Design and Implementation of a Multiagent Planning Simulation System

By

Thomas Kwan

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

University of Manitoba

Winnipeg, Manitoba

October, 1993

ISBN 0-315-85926-1

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

COMPUTER SCIENCE
SUBJECT TERM

0984
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology ... 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology .. 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics ... 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Nom _____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

_____ ☐☐☐ U·M·I

SUJET — CODE DE SUJET

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS
Architecture 0729
Beaux-arts 0357
Bibliothéconomie 0399
Cinéma 0900
Communication verbale 0459
Communications 0708
Danse 0378
Histoire de l'art 0377
Journalisme 0391
Musique 0413
Sciences de l'information 0723
Théâtre 0465

ÉDUCATION
Généralités 515
Administration 0514
Art 0273
Collèges communautaires 0275
Commerce 0688
Économie domestique 0278
Éducation permanente 0516
Éducation préscolaire 0518
Éducation sanitaire 0680
Enseignement agricole 0517
Enseignement bilingue et multiculturel 0282
Enseignement industriel 0521
Enseignement primaire. 0524
Enseignement professionnel 0747
Enseignement religieux 0527
Enseignement secondaire 0533
Enseignement spécial 0529
Enseignement supérieur 0745
Évaluation 0288
Finances 0277
Formation des enseignants 0530
Histoire de l'éducation 0520
Langues et littérature 0279
Lecture 0535
Mathématiques 0280
Musique 0522
Orientation et consultation 0519
Philosophie de l'éducation 0998
Physique 0523
Programmes d'études et enseignement 0727
Psychologie 0525
Sciences 0714
Sciences sociales 0534
Sociologie de l'éducation 0340
Technologie 0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE
Langues
- Généralités 0679
- Anciennes 0289
- Linguistique 0290
- Modernes 0291

Littérature
- Généralités 0401
- Anciennes 0294
- Comparée 0295
- Médiévale 0297
- Moderne 0298
- Africaine 0316
- Américaine 0591
- Anglaise 0593
- Asiatique 0305
- Canadienne (Anglaise) 0352
- Canadienne (Française) 0355
- Germanique 0311
- Latino-américaine 0312
- Moyen-orientale 0315
- Romane 0313
- Slave et est-européenne 0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE
Philosophie 0422
Religion
- Généralités 0318
- Clergé 0319
- Études bibliques 0321
- Histoire des religions 0320
- Philosophie de la religion 0322

Théologie 0469

SCIENCES SOCIALES
Anthropologie
- Archéologie 0324
- Culturelle 0326
- Physique 0327

Droit 0398
Économie
- Généralités 0501
- Commerce-Affaires 0505
- Économie agricole 0503
- Économie du travail 0510
- Finances 0508
- Histoire 0509
- Théorie 0511

Études américaines 0323
Études canadiennes 0385
Études féministes 0453
Folklore 0358
Géographie 0366
Gérontologie 0351
Gestion des affaires
- Généralités 0310
- Administration 0454
- Banques 0770
- Comptabilité 0272
- Marketing 0338

Histoire
- Histoire générale 0578
- Ancienne 0579
- Médiévale 0581
- Moderne 0582
- Histoire des noirs 0328
- Africaine 0331
- Canadienne 0334
- États-Unis 0337
- Européenne 0335
- Moyen-orientale 0333
- Latino-américaine 0336
- Asie, Australie et Océanie 0332

Histoire des sciences 0585
Loisirs 0814
Planification urbaine et régionale 0999
Science politique
- Généralités 0615
- Administration publique 0617
- Droit et relations internationales 0616

Sociologie
- Généralités 0626
- Aide et bien-être social 0630
- Criminologie et établissements pénitentiaires 0627
- Démographie 0938
- Études de l' individu et de la famille 0628
- Études des relations interethniques et des relations raciales 0631
- Structure et développement social 0700
- Théorie et méthodes. 0344
- Travail et relations industrielles 0629

Transports 0709
Travail social 0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES
Agriculture
- Généralités 0473
- Agronomie. 0285
- Alimentation et technologie alimentaire 0359
- Culture 0479
- Élevage et alimentation 0475
- Exploitation des pâturages 0777
- Pathologie animale 0476
- Pathologie végétale 0480
- Physiologie végétale 0817
- Sylviculture et faune 0478
- Technologie du bois 0746

Biologie
- Généralités 0306
- Anatomie 0287
- Biologie (Statistiques) 0308
- Biologie moléculaire 0307
- Botanique 0309
- Cellule 0379
- Écologie 0329
- Entomologie 0353
- Génétique 0369
- Limnologie 0793
- Microbiologie 0410
- Neurologie 0317
- Océanographie 0416
- Physiologie 0433
- Radiation 0821
- Science vétérinaire 0778
- Zoologie 0472

Biophysique
- Généralités 0786
- Medicale 0760

SCIENCES DE LA TERRE
Biogéochimie 0425
Géochimie 0996
Géodésie 0370
Géographie physique 0368
Géologie 0372
Géophysique 0373
Hydrologie 0388
Minéralogie 0411
Océanographie physique 0415
Paléobotanique 0345
Paléoécologie 0426
Paléontologie 0418
Paléozoologie 0985
Palynologie 0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT
Économie domestique 0386
Sciences de l'environnement 0768
Sciences de la santé
- Généralités 0566
- Administration des hôpitaux 0769
- Alimentation et nutrition 0570
- Audiologie 0300
- Chimiothérapie 0992
- Dentisterie 0567
- Développement humain 0758
- Enseignement 0350
- Immunologie 0982
- Loisirs 0575
- Médecine du travail et thérapie 0354
- Médecine et chirurgie 0564
- Obstétrique et gynécologie 0380
- Ophtalmologie 0381
- Orthophonie 0460
- Pathologie 0571
- Pharmacie 0572
- Pharmacologie 0419
- Physiothérapie 0382
- Radiologie 0574
- Santé mentale 0347
- Santé publique 0573
- Soins infirmiers 0569
- Toxicologie 0383

SCIENCES PHYSIQUES
Sciences Pures
Chimie
- Généralités 0485
- Biochimie 487
- Chimie agricole 0749
- Chimie analytique 0486
- Chimie minérale 0488
- Chimie nucléaire 0738
- Chimie organique 0490
- Chimie pharmaceutique 0491
- Physique 0494
- PolymÈres 0495
- Radiation 0754

Mathématiques 0405
Physique
- Généralités 0605
- Acoustique 0986
- Astronomie et astrophysique 0606
- Électronique et électricité 0607
- Fluides et plasma 0759
- Météorologie 0608
- Optique 0752
- Particules (Physique nucléaire) 0798
- Physique atomique 0748
- Physique de l'état solide 0611
- Physique moléculaire 0609
- Physique nucléaire 0610
- Radiation 0756

Statistiques 0463

Sciences Appliqués Et Technologie
Informatique 0984
Ingénierie
- Généralités 0537
- Agricole 0539
- Automobile 0540
- Biomédicale 0541
- Chaleur et thermodynamique 0348
- Conditionnement (Emballage) 0549
- Génie aérospatial 0538
- Génie chimique 0542
- Génie civil 0543
- Génie électronique et électrique 0544
- Génie industriel 0546
- Génie mécanique 0548
- Génie nucléaire 0552
- Ingénierie des systèmes 0790
- Mécanique navale 0547
- Métallurgie 0743
- Science des matériaux 0794
- Technique du pétrole 0765
- Technique minière 0551
- Techniques sanitaires et municipales 0554
- Technologie hydraulique 0545

Mécanique appliquée 0346
Géotechnologie 0428
Matières plastiques (Technologie) 0795
Recherche opérationnelle 0796
Textiles et tissus (Technologie) 0794

PSYCHOLOGIE
Généralités 0621
Personnalité 0625
Psychobiologie 0349
Psychologie clinique 0622
Psychologie du comportement 0384
Psychologie du développement 0620
Psychologie expérimentale 0623
Psychologie industrielle 0624
Psychologie physiologique 0989
Psychologie sociale 0451
Psychométrie 0632

DESIGN AND IMPLEMENTATION OF A MULTIAGENT PLANNING SIMULATION SYSTEM

BY

THOMAS KWAN

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Abstract

Multiagent planning systems coordinate a number of agents to solve problems. Each agent contributes its expertise in a certain area so that a complicated problem can be solved by the group of agents cooperatively. This thesis discusses the design and implementation of a multiagent planning system. An examination of single and multiagent planning techniques is presented as is a survey of various systems developed to date. The components of our system, how they are implemented, and how they are used are also examined. Samples of the application of the system to an actual problem are then presented.

Acknowledgments

I thank God for his blessing in the past years of my study. To God be the glory.

I would especially like to thank my wife Ivy Au in giving me full support of my study. Her support behind my work gives me tremendous strength in completing it.

I would like to thank my supervisor, Dr. Mark Evans, for his leadership, guidance, and abundance of ideas. Without him, this thesis would not have been possible.

I would also like to thank Dr. David Scuse and Dr. Dean Kriellaars for sitting on my committee.

Lastly, I would like to thank my parents for all the support and guidance they have given me over the years of my life.

TABLE OF CONTENTS

Abstract i
Acknowledgments ii

1. **Introduction** 1
1.1 History of Planning Research 1
1.2 Classical Planning 2
1.3 Multiagent Planning 4
1.4 Simulation of Multiagent Planning System 5

2. **Single Agent Planning** 8
2.1 Introduction to Single Agent Planning 8
2.2 STRIPS 8
2.2.1 History 8
2.2.2 Assumptions made 9
2.2.3 Problem Space 9
2.2.4 Operations 10
2.2.5 Example 12
2.2.6 Summary 16
2.3 NONLIN 17
2.3.1 Introduction 17
2.3.2 Assumptions made 18
2.3.3 Backtracking Scheme 18
2.3.4 Task Schema 19
2.3.5 Plan Critics 20
2.3.6 Question Answer Facility 23
2.3.7 Summary 25
2.4 Other Single Agent Planners 25
2.4.1 ABSTRIPS 25
2.4.2 NOAH 26
2.5 Summary 26

3. **Distributed Artificial Intelligence** 28
3.1 Introduction 28
3.2 Considerations in DAI 29
3.2.1 Description, Decomposition, and Allocation of Tasks 29
3.2.2 Interaction, Language, and Communication 30
3.2.3 Coherence and Coordination 31
3.2.4 Modeling Other Agents and Organized Activity 31
3.2.5 Disparity and Conflict 32
3.3 Blackboard Systems 32
3.3.1 Introduction 32
3.3.2 Framework 33

3.4 Contract Networks 35
3.4.1 Introduction 35
3.4.2 Contract-Net Protocol 35
3.4.3 Example Application 36
3.4.4 Contract-Net Summary 39
3.5 Summary 40

4. **Multiagent Planning** 41
4.1 Introduction 41
4.2 Knowledge-based Model of an Agent 42
4.2.1 Basic Components 42
4.2.2 Constraints 43
4.2.3 Communication Structures 44
4.2.4 Knowledge Sources 45
4.2.5 Activity Blackboard 46
4.3 Constraint-directed Planning 47
4.3.1 Planning Control Strategy 47
4.3.2 Plan Evaluation 48
4.3.3 Constraint Relaxation 48
4.4 Basic Functions 49
4.4.1 Task Decomposition 50
4.4.2 Interaction and Communication 51
4.5 Summary 52

5. **Implementation** 53
5.1 Data Structures 53
5.1.1 Plan Structure 54
5.1.2 Task Structure 55
5.1.3 Agent Class Structure 56
5.1.4 Agent Structure 56
5.1.5 Request Structure 58
5.1.6 Notification Structure 60
5.1.7 Activity Structure 61
5.2 Program Structure 64
5.2.1 Basic Algorithm 64
5.2.2 Process Requests 66
5.2.3 Process Activity Nodes 67
5.2.4 Process Notifications 73
5.3 Summary 74

6. **Sample System Runs** 75
6.1 Agent Knowledge 75
6.2 Sample Runs 76
6.2.1 Dataset A 76
6.2.2 Dataset B1 85

6.2.3 Dataset B2 86
6.2.4 Dataset C1 89
6.2.5 Dataset D1 91
6.2.6 Dataset D2 94
6.3 Summary 96

7. **Conclusions** 97

Appendix A : References 100

Appendix B : Data Structures and Program Routines 103

Appendix C : Testing Data and Sample Runs 128

Chapter One : Introduction

1.1 History of Planning Research

Reasoning about actions is a fundamental characteristic of intelligent behavior. Most every action we take in our daily life has some kind of reasoning behind it. For example, when one wants to buy a drink, it may be because he or his wife is thirsty. After we have decided to achieve a goal, quite often there are many different ways to accomplish the goal. Each possibility usually includes a set of actions (a plan) instead of one action. For instance, one may need to get some money and drive a few miles before he can buy a drink. Decades of research in Artificial Intelligence (AI) and related disciplines have shown that such human capability is extremely difficult to formalize but it is an essential component of intelligent behavior. In an automated planning system, the set of actions derived to attain a goal is often passed to a robot or a manufacturing system, which can follow the plan and produce the desired result. The design of such *planners* has been one of the major research topics since AI research's earliest days, and many techniques have been introduced. Like most AI research, planning problems have been attacked in two major ways [Wilkins, 1988]. The first approach is try to understand and solve the general problem without the use of domain-specific knowledge. The second approach incorporates large amounts of domain-specific knowledge. These approaches are often called domain independent and domain dependent respectively. Domain dependent refers to those systems that use domain-specific heuristics (knowledge) to control the planner's operations. Domain independent refers to those systems in which the planner's knowledge representation and planning algorithms are expected to work for a reasonably large variety of application domains. Domain dependent planners are generally found in applied approaches in AI where the design principles might not map well from one application domain to another. However, work in domain independent planning system has formed a large area in AI research.

A problem being solved by a planner is characterized by an initial state and a goal state description. The initial state describes the way the world is before planning begins. The goal state describes the way we want the world to look when a plan has been designed and executed. The world refers to the application domain where the planning takes place. The planner tries to find a plan (a set of actions) such that by executing the plan from the initial state of the problem, the goal state will be generated. Planning is essentially a search problem. The planner must traverse a potentially large search space and find a plan that is applicable in the initial state and produces the goal state when carried out. The complexity of the planning system grows when more than one applicable plan is found. In these cases, the most applicable plan must be chosen from among all the possible plans.

1.2 Classical Planning

The classical definition of the planning problem assumes a state-based representation of the world [Wilkins, 1988]. The world is represented by taking a "snapshot" of it at one time and describing the world as it appears in this snapshot. The planner assumes that the initial state of the world does not change while the planner is executing. Thus, the planner constructs a plan based on the initial state of the world without reacting to the changing state of the world during the planning process. This is an important limitation in classical planners. This also leads to the distinction between plan time and execution time. None of the classical planners can react to changes in the world while the system is generating a plan. Examples of such classical planners are STRIPS and NONLIN [Morgenstern, 1988].

Figure 1 shows a classical planning model [Wilkins, 1988]. It describes the production of a sequence of operators that alter the environment in such a way as to accomplish a goal. The representation of the model performs a mapping from the real world to the planner.

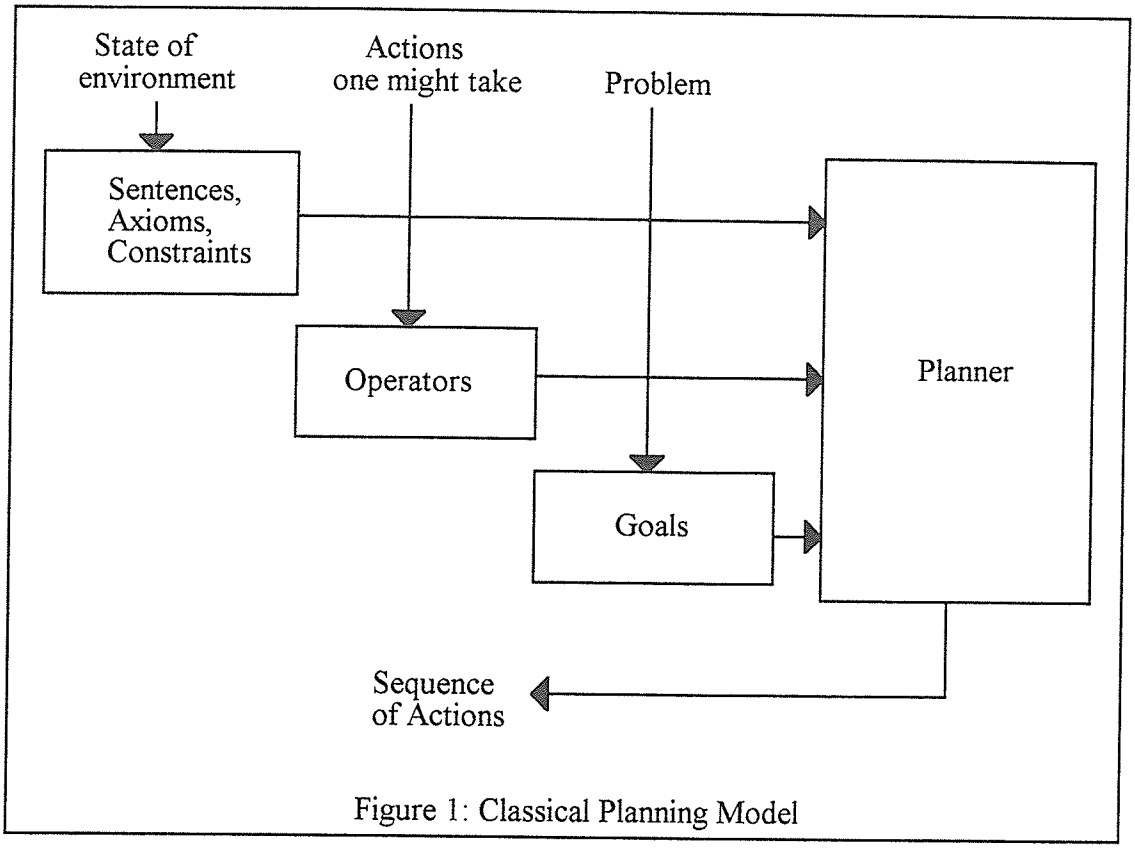


Figure 1: Classical Planning Model

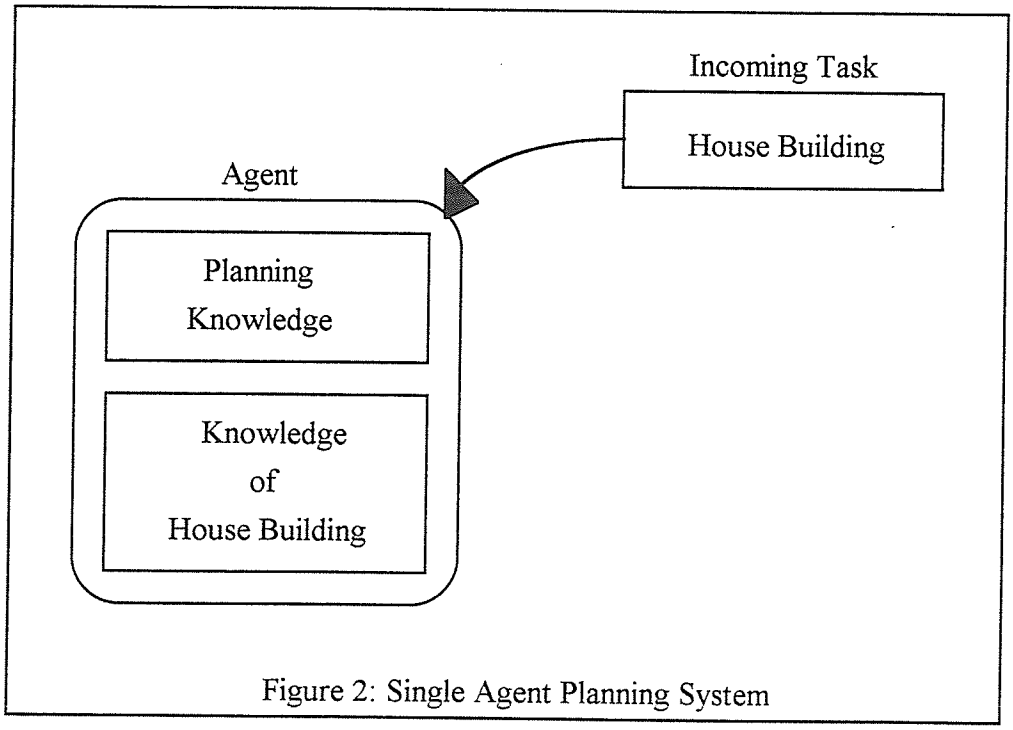


Figure 2: Single Agent Planning System

In classical planning, one agent does all the planning work. The agent must have all the required knowledge specified in the application domain to construct applicable plans. Figure 2 illustrates that an agent must have all the required knowledge of building a house to complete a task of building a house from a given set of specifications (initial state and goal state). This is often not true in real world problems. In an organization, a group of people can accomplish things that individuals cannot. In this situation, different agents must cooperate with one another and contribute their expertise to solve a problem. This leads to a research area in AI known as Distributed Artificial Intelligence which concentrates on the development of multiagent planning systems.

1.3 Multiagent Planning

Multiagent planning research studies how a loosely coupled network of problem solvers (or agents) can work together to solve problems that are beyond their individual capabilities [Durfee, Lesser and Corkill, 1989]. Each agent in the network is capable of solving sophisticated problems using its own expertise and can work independently. Many problems faced by the agents, however, cannot be completed without their cooperation. Cooperation is necessary because no single agent has enough expertise, resources, and information to solve a particular problem. Agents must rely on each other to solve the problem. For example, if the problem is to build a house, one agent might have expertise on designing the appearance of the house; one agent might have expertise on laying the foundation of the house; another agent on plumbing; and so on. Figure 3 shows how an agent can divide an incoming house building task into sub-tasks and send some subtasks to other agents. This agent only has the knowledge of building a house frame. The other required work such as laying the foundation, plumbing and decoration are sent to other agents. The planning component of the agent provides the knowledge of dividing the incoming house building task into sub-tasks.

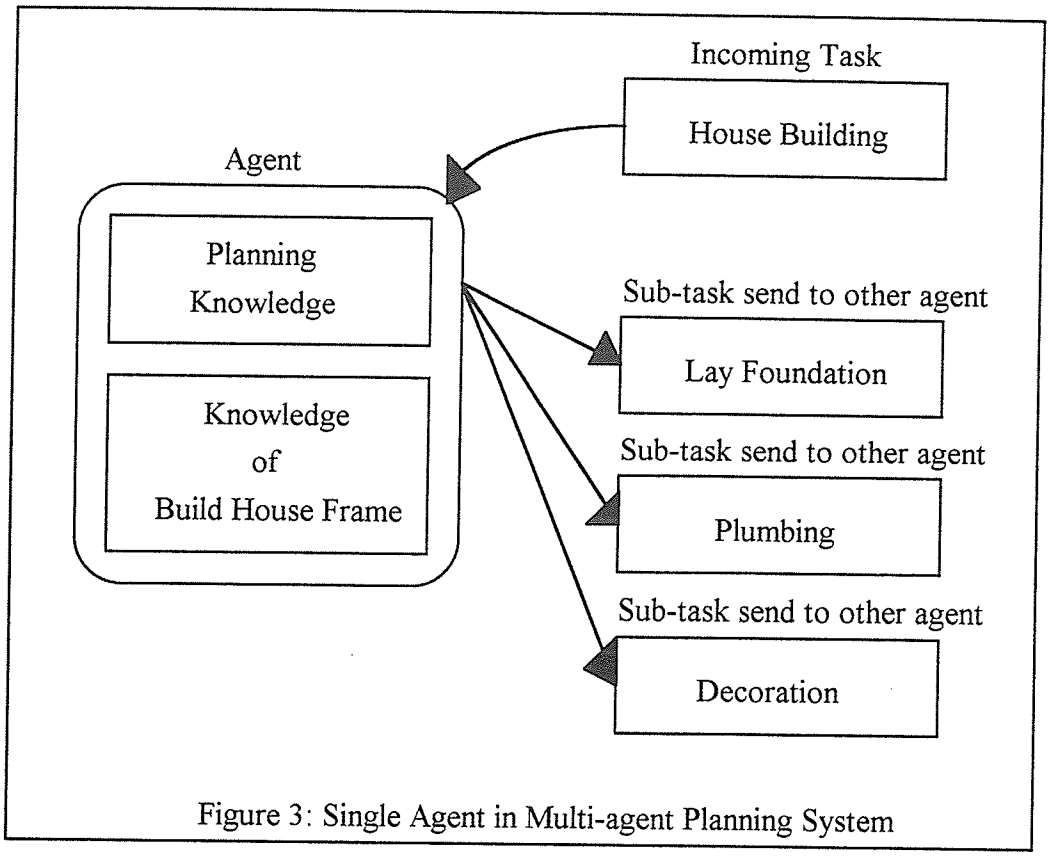


Figure 3: Single Agent in Multi-agent Planning System

In multiagent planning systems, agents solve a problem cooperatively by using their local expertise, resources and information to solve a set of sub-problems individually, and then integrate these sub-problem solutions into a complete solution.

1.4 Simulation of Multiagent Planning Systems

The objective of this thesis is to create a test bed for a general purpose multiagent planning system. Agents in the system contain all the required basic components. Figure 4 shows the major components of a basic agent [Evans and Anderson, 1989].

The planning component contains the knowledge necessary to divide incoming tasks into sub-tasks, coordinate the activities and integrate the results into a final solution. The problem solving component contains the knowledge representing the agent's own

capability. Instead of physically creating the agents, they are simulated as independent processes in a computer system. All the activities of an agent including planning, problem solving and communications between the agents are also simulated. The result is a simulation tool that can be used to test and evaluate various methods for creating multiagent planning systems.

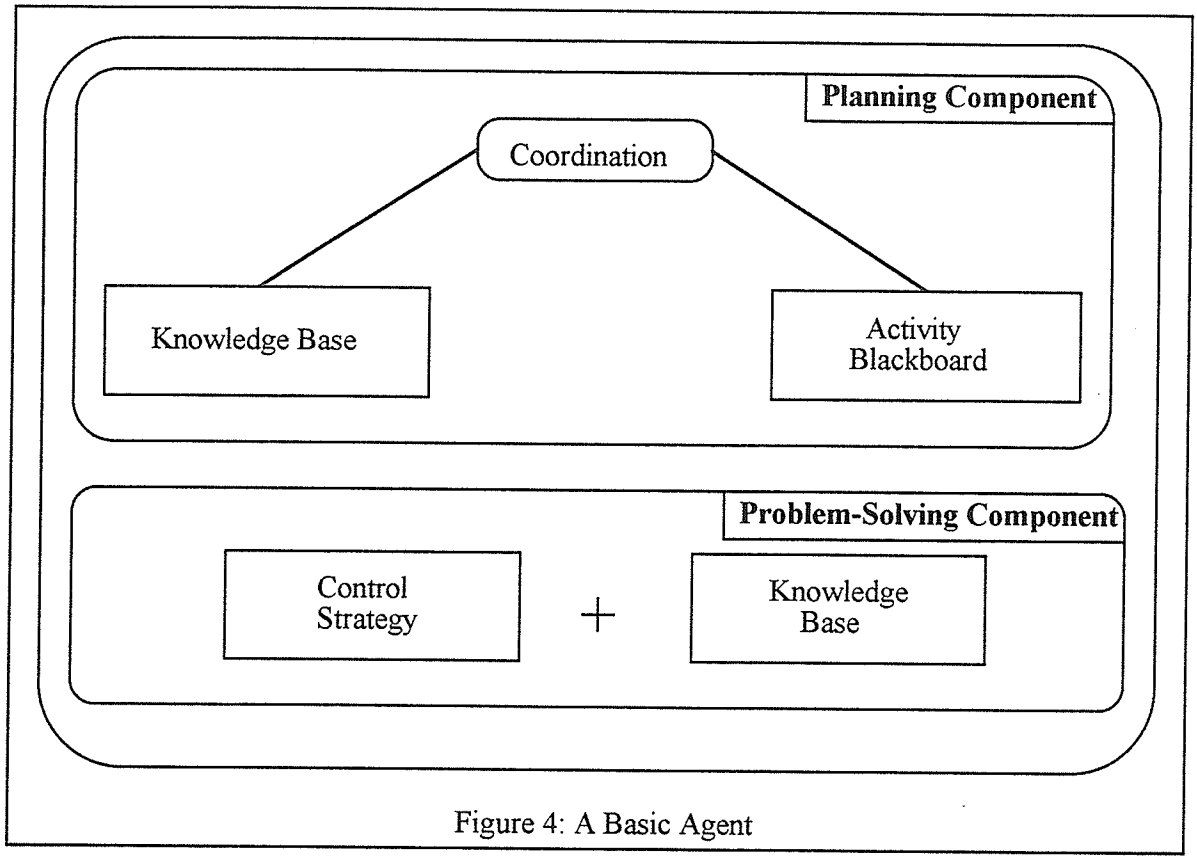


Figure 4: A Basic Agent

House building is the example problem that has been used to illustrate the application of the system for this thesis. The sub-problems generated by the agents do not completely describe the actual processes involved in building a house. Only certain major processes are chosen and built into the knowledge bases of the agents. The system can be expanded to include other processes as well. The knowledge component of the system is separated from the planning engine making the system easy to modify for other problems.

The system was developed and tested in Macintosh Common LISP. The simulated environment provides instant access to the components of the agents before, during, and after planning. This enables the developer to trace and examine the operation of each agent as it contributes to the planning process. Various options are also available to control the operation of the system (see Chapter 6). The system runs on a Macintosh computer. Other platforms with compatible LISP environments can be used as well.

Chapter Two : Single Agent Planning

2.1 Introduction to Single Agent Planning

Classical planners are single agent planning systems. Problems that can be solved by the planner are limited by the agent's domain knowledge. The agent must have a complete, correct description of the world and it is the only entity that can affect the environment [Barr, Cohen and Feigenbaum, 1989]. By examining the initial state of the world, the agent constructs a plan based on its domain knowledge. After applying the plan to the initial state, the goal state is achieved. The agent itself must be capable of performing all the required actions to achieve the goal. If there is one action that the agent fails to complete and there is no alternative plan for it, then the ultimate goal will not be completed. The entire job must be aborted.

A computer processor has a limited processing capacity. A processor can execute only a limited number of instructions per second. This limits the amount of information a processor may process and the amount of control it may exercise within a given time period. Hence, programmed systems, whether centralized or distributed, may exhibit symptoms similar to the bounded rationality exhibited by humans when capacities are exceeded[1]. The metaphor appears viable [Fox, 1988]. In this case, a single agent planing system can only solve relatively simple problem. Several assumptions are made by classical planners. The planner always has a complete, correct description of the world. The planner is the only entity that can affect the environment - nothing happens spuriously and no other agents can affect the world. The planner can spend as much time planning as needed. The plans will always execute correctly. In this chapter we examine some significant

[1] *Bounded rationality* means that the capacity of the human mind for formulating and solving complex problems is very small compared with the size of the problems whose solution is required for objectively rational behavior in the real world - or even for a reasonable approximation to such objective rationality [Fox, 1988].

classical planning systems. Further information on classical planning is found in [Wilkins, 1988].

2.2 STRIPS

2.2.1 History

STRIPS (STanford Research Institute Problem Solver) is a single agent planning system developed by Richard Fikes and Nils Nilsson (1971) at SRI International [Fikes and Nilsson, 1971]. Each problem for STRIPS is a goal to be achieved by a robot. The world in which the STRIPS robot works consists of several rooms connected by doors, along with some boxes and other objects that the robot can manipulate. Operators are the basic elements from which a solution is built. For robot problems, each operator corresponds to an action performed by the robot. Typical operators include going to a specific location and pushing an object to a location. The locations are given as parameters.

2.2.2 Assumptions made

The initial world model defined in STRIPS is created by making a snapshot of the current world. Once the snapshot is made, STRIPS assumes that it will not be changed. No other object can possibly affect the world model while STRIPS creates the plan. STRIPS also assumes that the operators in the system will be performed properly. For example, when a robot arm picks up a box, it must pick it up without error. If it slips, the world model will be changed unpredictably and the entire plan may not work properly.

2.2.3 Problem Space

The problem space for STRIPS is defined by the initial world model, the set of available operators and their effects on world models, and the goal to be achieved. STRIPS represents a world model by a set of well-formed formulas (wffs) [Fikes and

Nilsson, 1971]. For example, to describe a world model in which the robot is at location a and boxes B and C are at location b and c, we would include the following wffs[2]:

$$\begin{aligned} &\text{ATR}(a) \\ &\text{AT}(B,b) \\ &\text{AT}(C,c) \end{aligned}$$

The operators are grouped into families called schemata. Consider the operator *goto* for moving the robot from one point to another. It is a distinct operator for each different pair of points, but it is convenient to group all of these into a family goto (m,n) parameterized by the initial position m and the final position n. Goto (m,n) is an operator schema whose members are obtained by substituting specific constants for the parameters m and n. Each operator consists of two main parts: a description of the effects of the operators, and the preconditions to its applicability [Fikes and Nilsson, 1971]. The effects of an operator are simply defined by a list of wffs that must be added to the world model when the operator is applied, and a list of wffs that are no longer true and therefore must be deleted. It is convenient to state the precondition for an operator schema as a wff schema. To determine whether there is an instance of an operator schema applicable to a world model, we must be able to prove that there is an instance of the corresponding wff schema that logically follows from the model.

A typical operator is push (k,m,n), which denotes the robot pushes object k from m to n. Such operator might be described as follows:

push(k,m,n)

Precondition: $\text{ATR}(m) \wedge \text{AT}(k,m)$

Delete List: $\text{ATR}(m)$
$\text{AT}(k,m)$

Add List: $\text{ATR}(n)$
$\text{AT}(k,n)$

---

[2]ATR(x) represents the fact that the robot is at location x.
ATR(Y,z) represents the object Y is at location z.

The precondition statement requires that the robot and the object k must be at location m. After applying operator push (k,m,n), the robot and the object k will no longer be at location m. That is why wffs ATR(m) and AT(k,m) are included in the delete list. On the other hand, the new location of the robot and the object k will be at n. That is why wffs ATR(n) and AT(k,n) are included in the add list.

2.2.4 Operations

STRIPS operates by searching a space of world models to find one in which the given goal is achieved. It uses a *state-space representation* in which each state is a pair consisting of a world model and a list of goals to be achieved [Fikes and Nilsson, 1971]. The initial state is $(M_0, (G_0))$, where M_0 is the initial world model and G_0 is the given goal. STRIPS begins by employing a theorem prover to attempt to prove that the goal wff G_0 follows from the set M_0 of wffs describing the initial world model. If G_0 does follow from M_0, the task is trivially solved in the initial model. Otherwise, the theorem prover will fail to find a proof. In this case, the uncompleted proof is taken to be the difference between M_0 and G_0. Next, operators that might be relevant to reducing this difference are sought. These are the operators whose effects on world models would enable the proof to be continued. In determining relevance, the parameters of the operators may be partially or fully instantiated. The corresponding instantiated precondition wff schemata are then taken to be new sub-goals.

STRIPS works on a sub-goal using the same technique. Suppose the precondition wff schema G_1 is selected as the first sub-goal to be worked on. STRIPS again uses a theorem prover in an attempt to find instances of G_1 that follow from the initial world model M_0. Here again, there are two possibilities. If no proof can be found, STRIPS uses the incomplete proof as a difference, and sets up sub-sub-goals corresponding to their precondition wffs. If STRIPS does find an instance of G_1 that follows from M_0, then the corresponding operator instance is used to transform M_0 into a new world model M_1.

The hierarchy of goal, sub-goals and models generated by the search process is represented by a *search tree*. Each node of the search tree has the form (<world model>, <goal list>), and represents the problem of trying to achieve the sub-goals on the goal list from the indicated world model. STRIPS uses the GPS[3] strategy of attempting to apply those operators that are relevant to reducing a difference between a world model and a goal or sub-goal. Theorem prover is the key part of this mechanism.

2.2.5 Example

To explore how STRIPS works, consider the following problem. Figure 5 a) and b) shows the initial state and goal state respectively.

Initial State

ON(B,A) ∧
ONTABLE(A) ∧
CLEAR(B) ∧
ARMEMPTY

Goal State

ON(A,B) ∧
ONTABLE(B) ∧
CLEAR(A) ∧
ARMEMPTY

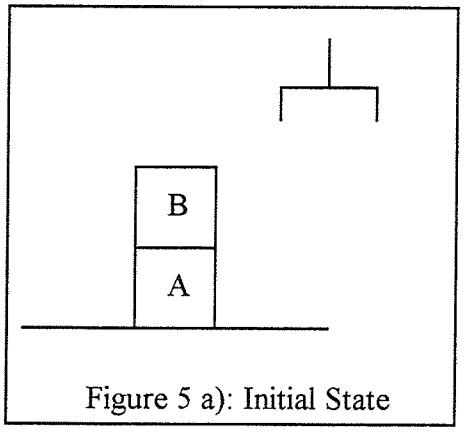


Figure 5 a): Initial State

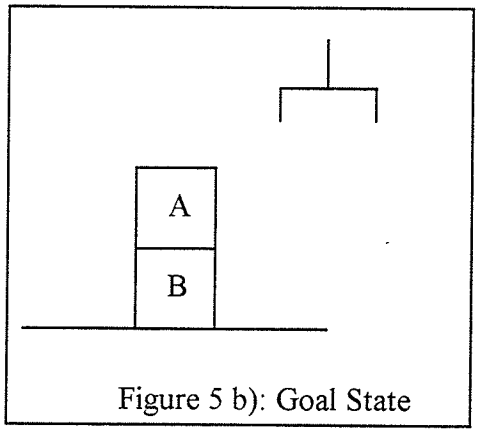


Figure 5 b): Goal State

Assume the planner is given the following operators:

STACK(x,y):
Precondition: CLEAR(y) ∧ HOLDING(x)
Delete List: CLEAR(y) ∧ HOLDING(x)
Add List: ARMEMPTY ∧ ON(x,y) ∧ CLEAR(x)

[3]GPS General purpose Problem Solver developed by Simon and Newell [Ernst and Newell, 1969].

UNSTACK(x,y):

Precondition:	ON(x,y) ∧ CLEAR(x) ∧ ARMEMPTY
Delete List:	ON(x,y) ∧ ARMEMPTY ∧ CLEAR(x)
Add List:	HOLDING(x) ∧ CLEAR(y)

PICKUP(x):

Precondition:	CLEAR(x) ∧ ONTABLE(x) ∧ ARMEMPTY
Delete List:	ONTABLE(x) ∧ ARMEMPTY ∧ CLEAR(x)
Add List:	HOLDING(x)

PUTDOWN(x):

Precondition:	HOLDING(x)
Delete List:	HOLDING(x)
Add List:	ONTABLE(x) ∧ ARMEMPTY ∧ CLEAR(x)

Initially, the goal stack[4] contains:

ONTABLE(B) ∧ ON(A,B)

The planner separates the problem into two subproblems. This leaves two possible goal stacks (depending on how we order the subgoals):

ONTABLE(B)	ON(A,B)
ON(A,B)	ONTABLE(B)
ONTABLE(B) ∧ ON(A,B)	ONTABLE(B) ∧ ON(A,B)

The planner must choose one of the goal stacks to process first. Let us assume that the left goal stack is used. The planner operates by pursuing the top goal on the stack. When a sequence of operators that solve it is found (zero or more), that sequence is applied to the current state to create a new state. The goal is then popped from the stack and the next goal is pursued starting from the new state. This process continues until the goal stack is empty.

The first thing the planner must do is to check whether ONTABLE(B) is true in the current state. It is not, therefore the planner checks for operators that can cause it to become true. Of our four operators, only PUTDOWN(B) can be used since its Add List

[4]The goal stack is used to enable the planner to keep track of the goals and subgoals it should pursue in order to solve a given problem.

contains ONTABLE(x). The planner then replaces ONTABLE(B) on the goal stack with PUTDOWN(B) producing:

PUTDOWN(B)
ON(A,B)
ONTABLE(B) ∧ ON(A,B)

However, to apply PUTDOWN(B), the planner must ensure that its preconditions are satisfied. This is done by adding the preconditions as sub-goals on the goal stack:

HOLDING(B)
PUTDOWN(B)
ON(A,B)
ONTABLE(B) ∧ ON(A,B)

The planner must check to see if HOLDING(B) is true. It finds that this goal is not true and therefore looks for operators that can make it true. UNSTACK(B,A) and PICKUP(B) are two possible operators. Lets say the planner chooses UNSTACK(B,A). The goal stack becomes:

ON(B,A)
CLEAR(B)
ARMEMPTY
UNSTACK(B,A)
PUTDOWN(B)
ON(A,B)
ONTABLE(B) ∧ ON(A,B)

The planner finds that the top three conditions of the goal stack are true, so they are removed from the goal stack. The action UNSTACK(B,A) is then performed as shown in Figure 6 (this modifies the world model).

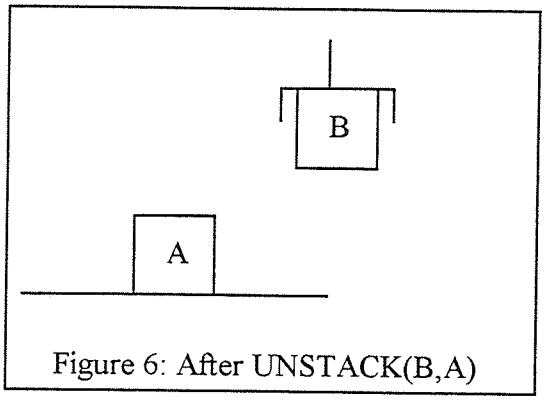


Figure 6: After UNSTACK(B,A)

After that, the next action in the goal stack, PUTDOWN(B), is also performed as shown in Figure 7.

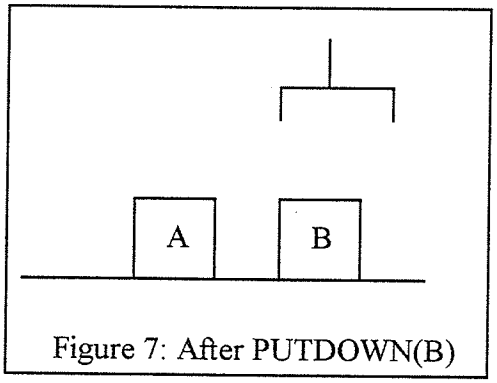


Figure 7: After PUTDOWN(B)

This leaves the goal stack as follows:

ON(A,B)
ONTABLE(B) ∧ ON(A,B)

The planner checks the first condition in the goal stack, ON(A,B), and finds that it is not true. ON(A,B) is replaced by the operator STACK(A,B). The goal stack becomes:

HOLDING(A)
CLEAR(B)
STACK(A,B)
ONTABLE(B) ∧ ON(A,B)

The planner finds that HOLDING(A) is not true. HOLDING(A) is replaced by operator PICKUP(A). The goal stack becomes:

CLEAR(A)
ONTABLE(A)
ARMEMPTY
PICKUP(A)
CLEAR(B)
STACK(A,B)
ONTABLE(B) ∧ ON(A,B)

Since the top three conditions in the goal stack are true, the planner performs the PICKUP(A) action as shown in Figure 8.

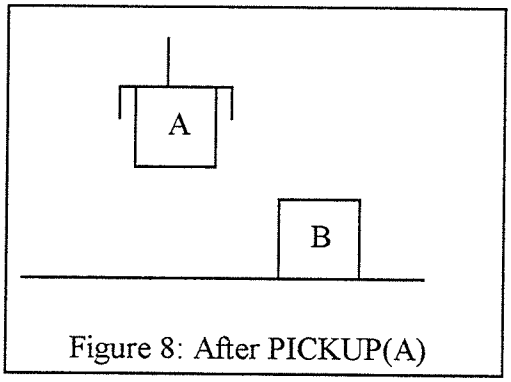


Figure 8: After PICKUP(A)

After that, since CLEAR(B) is also true, the planner performs the STACK(A,B) action as shown in Figure 9.

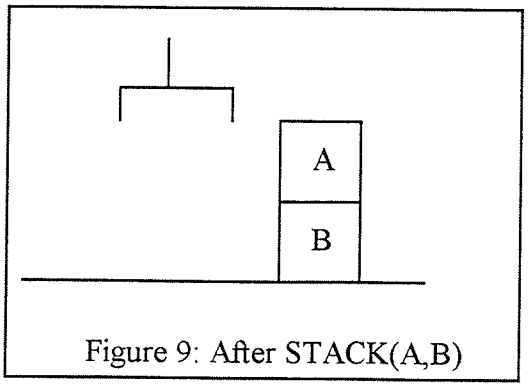


Figure 9: After STACK(A,B)

Finally, the planner checks the last condition in the goal stack and finds that it is true. Problem solved!

2.2.6 Summary

STRIPS constructs a problem-solving tree whose nodes represent sub-problems. In a problem-solving process of this sort, there must be a mechanism to decide which node to work on next. STRIPS uses an evaluation function that incorporates such factors as the number and the estimated difficulty of the remaining sub-goals; the cost of the operators applied so far; and the complexity of the current difference. Other evaluation functions and other ordering techniques can be introduced to the system.

2.3 NONLIN

2.3.1 Introduction

NONLIN is a classical single agent planning system. It generates a plan given a goal in a particular domain. Since the goals and sub-goals often interact with one another, it needs to drop the linearity assumption[5]. In allowing plans to be partially ordered, we need nonlinear plans. "NONLIN" stands for its nonlinear characteristics. A plan is nonlinear if it contains actions that are unordered regarding each other, i.e., the order has not yet been determined or actions may be executed in parallel [Tate, 1977]. Nonlinear planning avoids searching for all possible orderings of actions. Orderings are only inserted when information justifying them becomes available. Consequently, ordering of actions becomes an explicit activity of the planner. NONLIN must reason about interactions among planned actions (e.g., resource contention, undoing effects) and determine how to correct them through orderings or insertion of additional actions.

---

[5]*Linearity assumption* means whenever the planner is faced with the problem of achieving a pair of conjuctive subgoals, it assumes that they can be achieved independently.[Cohen and Feigenbaum, 1989]

NONLIN's approach varies dramatically from the state-spaced approach to planning employed by STRIPS. Nodes in the search space are partially ordered plans at some level of abstraction rather than world states. NONLIN changed the search problem in planning to a space of partial plans. For any non-primitive action in the network, NONLIN can consider any known method of reducing this action to a set of other actions or primitives. Planning consists of choosing appropriate reductions from the sets of possibilities and ordering actions to eliminate harmful interactions. Each state in the problem space represents a set of possible plans. The more abstract the state, the more possible plans that can be generated.

2.3.2 Assumptions made

Like STRIPS, the initial world model defined in NONLIN is created by making a snapshot of the current world. Once the snapshot is made, NONLIN assumes that it will not be changed. No other object can possibly affect the world model while NONLIN creates the plan. NONLIN also assumes that the operators in the system will be performed properly.

2.3.3 Backtracking Scheme

NONLIN includes provision for backtracking when choices made are proved to be wrong. To backtrack, the planning system simply saves the state (partial plan) of the solution at some choice point and the set of alternative choices. The system picks a choice from this set and the search continues. If failure occurs, the saved state at the most recent choice point is restored and the next alternative is taken. If there are no alternatives, backtracking continues to the next most recent choice point. Backtracking of this sort can be implemented using stack-based techniques [Tate, 1977]. The amount of information saved for a state is typically small in comparison to saving complete world states, although for large plans a fair amount of information may be involved. Since backtracking is based

on the chronological order in which decisions were made, it is a *Chronological* backtracking strategy. NONLIN used a variant of depth-first search. At each choice point, it ranks the alternatives, selects the highest ranked alternative and follows it. If the planner is forced to backtrack, it picks the next highest ranked alternative and follows it. This process can be applied continually until no further alternatives are available (at which point the planner must backtrack to another choice point).

2.3.4 Task Schema

Another major enhancement in NONLIN was the development of a declarative *Task Formalism* that enabled actions in the domain to be described in a hierarchical manner [Tate, 1977]. The knowledge-base of a NONLIN planner consists of a set of these Task Schemas, each of which describes a task at some level of abstraction. A task schema typically contains information of when to introduce an action in a plan, the effects of the action, the conditions that must hold before the action can be performed and the way to expand the action to lower level actions without having to specify all the details of the lower level actions. The structured syntax and clarity of the representation make it easier to specify planning knowledge. Task descriptions can be created as modular units that are referenced by other units when necessary. For example, a task schema may look like the following:

```
ACTSCHEMA SERVICE
   PATTERN <<Install Services>>

   EXPANSION
   1  ACTION  <<Install Drains>>
   2  ACTION  <<Lay Storm Drains>>
   3  ACTION  <<Install Rough Plumbing>>
   4  ACTION  <<Install Finished Plumbing>>
   5  ACTION  <<Install Rough Wiring>>
   6  ACTION  <<Finish Electrical Work>>
   7  ACTION  <<Install Kitchen Equipment>>
   8  ACTION  <<Install Air Conditioning>>
```

```
ORDERINGS 1 --> 3   3 --> 4   5 --> 6   3 --> 7   5 --> 7

CONDITIONS
SUPERVISED      <<Drains Installed>>            AT 3 FROM 1
SUPERVISED      <<Rough Plumbing Installed>>    AT 4 FROM 3
SUPERVISED      <<Rough Wiring Installed>>      AT 6 FROM 5
UNSUPERVISED    <<Foundations Laid>>            AT 1
UNSUPERVISED    <<Flooring Finished>>           AT 4
```

Figure 10 shows the dependencies between the actions specified in the task schema. The condition statements also reflect the proper order of the actions to be executed.

NONLIN supports the inclusion of three basic types of conditions in a task schema. SUPERVISED conditions correspond to preconditions for the actions specified in the expansion of the task. These directly influence orderings among the actions in the expansion list. UNSUPERVISED conditions must be true for the action to be carried out, but the task schema does not contain actions in the expansion list to satisfy these conditions. USE-WHEN or HOLDS conditions must hold before the action can be used. They determine the relevancy of the action in conjunction with the PATTERN associated with the action. Only SUPERVISED conditions can be expanded. The other conditions must be satisfied by other actions through linking the nodes appropriately.

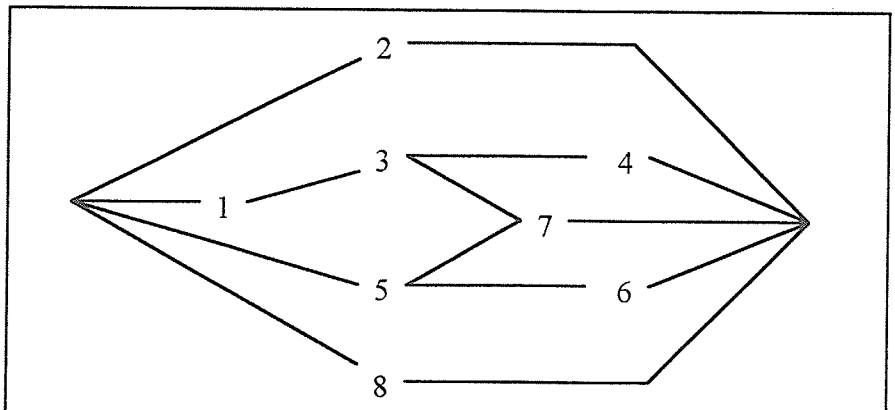


Figure 10: Dependencies of Actions

2.3.5 Plan Critics

NONLIN employs *plan critics* to check for interferences among the actions that are in a given plan at some level of abstraction [Tate, 1977]. Some of these critics are domain-independent while others may be domain-specific. These critics use a structure called the *Table of Multiple Effects (TOME)* [Tate, 1977]. This table is built for each level in the network and contains entries for expressions that were asserted or denied by more than one node in the current plan. The Resolve-Conflict critic examines portions of plan that are to be achieved in parallel. It uses the TOME to recognize when an expression that is asserted at some node is denied at a node that is not the asserting node's sub-goal.

Besides the TOME, NONLIN uses a GOAL STRUCTURE (GOST) [Tate, 1977]. A GOST is associated with each partial plan state and stores conditions on each node in the plan together with a list of contributors. Contributors are nodes that can make the condition hold. A GOST entry has the following structure:

[<condtype> <pattern> <value> <nodenum>] <list of contributors>

[SUPERVISED <<PLUMBING INSTALLED>> TRUE 6] with value [4]

In this example, the condition had to be true at node 6 and was made true at node 4 that is a contributor. The GOST allows the planner to determine the purposes of any particular effect at any node. The GOST entries specify a set of ranges for which patterns must have a certain value. The planner must try to protect these conditions for the required ranges. This allows interactions to be detected and allows corrections to be sensitive to the important effects of nodes.

A node in a plan is expanded to get more detail about how a task can be performed or a goal can be achieved. There are certain types of nodes that cannot be expanded. They are simply used to define the network of nodes and to define conditions that must hold but cannot be expanded. There are three types of nodes that are considered expandable. They are *Goal Node*, *Phantom Node* and *Action Node*. A goal node is present to state that the

pattern of the node should be true at the node. Phantom nodes represent goals that should already be true by the time they are met. An action node is present as a command to do something.

Figure 11 shows a conflict in a partial plan. The action 'Clear B' noted by a '+' sign conflicts with the action 'Put A on B'.

The Resolve-Conflict critic examines portions of the plan in Figure 11. It uses the TOME to recognize a conflict when an expression that is asserted at some node is denied at a node that is not the asserting node's sub-goal. The plan is modified (by enforcing an ordering on the nodes) to eliminate the conflict as shown in Figure 12.

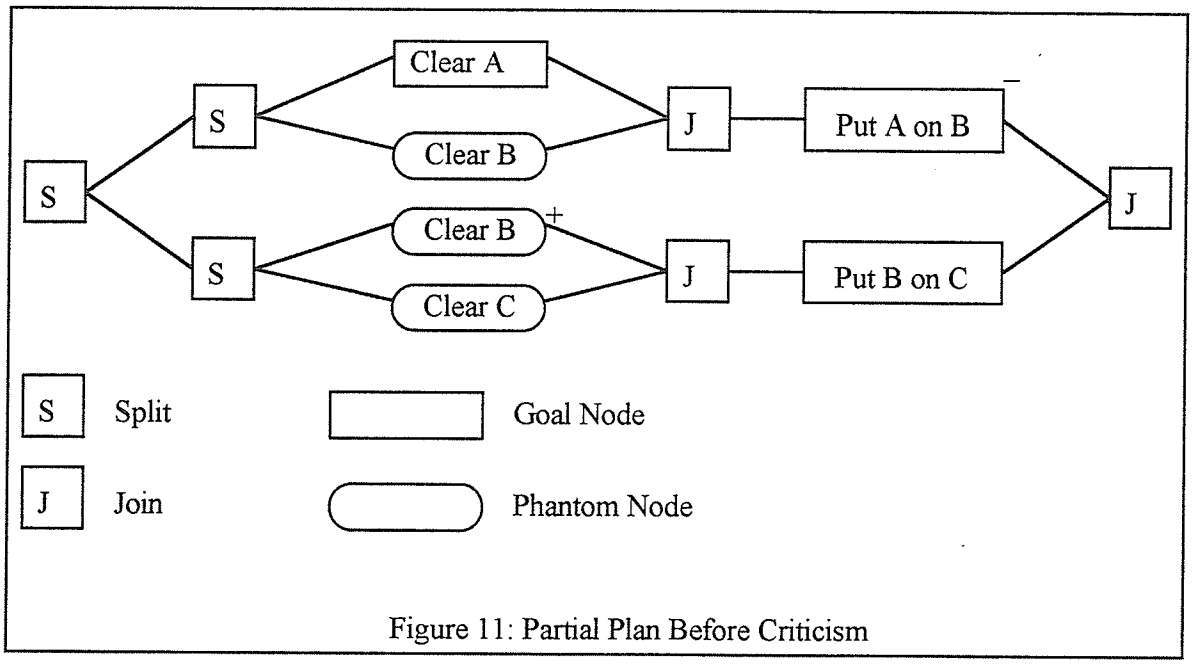


Figure 11: Partial Plan Before Criticism

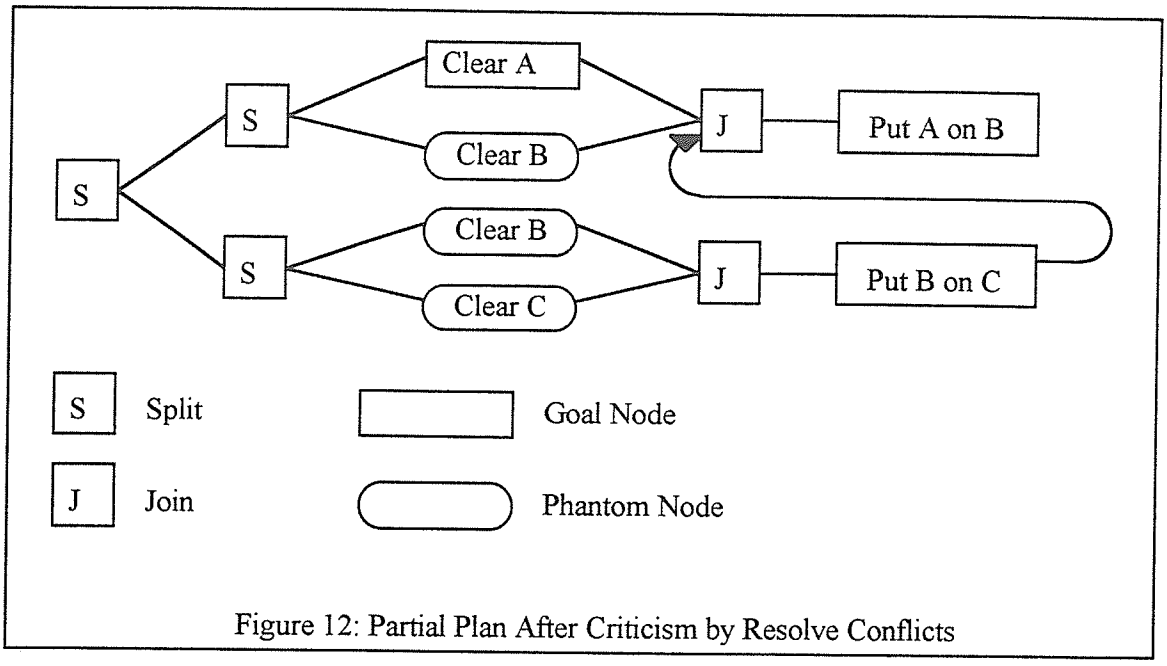


Figure 12: Partial Plan After Criticism by Resolve Conflicts

2.3.6 Question Answer Facility

NONLIN contains a question answering facility (QA) which is used by the planner to determine the value of a given statement in a given partially ordered plan [Tate, 1977]. The planner often needs to know what value a statement P will have at some node N in the plan. This is required to determine if a condition holds or if a goal is already satisfied. QA can respond to two types of queries.

a) Does statement P have a value V at node N in a plan?

b) What links would have to be added to the current plan to make P have a certain value at N?

Both queries involve creating a list of critical nodes regarding P and N. A P-node is a node that gives statement P a value. A PV-node is a node that gives statement P a value V. A P~V-node is a node that gives statement P a value other than V. A critical node for (P,N) is a node that gives a value to statement P that can be maintained up to node N. Critical nodes are the last P-node on each incoming branch to N and all P-nodes that are in parallel with N. QA(P,V,N) finds the list of critical nodes by marking the other nodes of the network with their position with respect to N and looking for TOME entries for P. P

definitely has a value V at N if there is at least one critical PV-node before N and there are no critical P~V-nodes. P definitely does not have value V at node N if there is at least one critical P~V-node and there are no critical PV-nodes. If neither of these conditions holds, then it may be possible to make P have a value V at N by adjusting the links among the nodes. The planner must ensure that at least one critical PV-node is linked in before N and that all critical P~V-nodes are linked out after N. This includes nodes that are in parallel with N. This task can be very difficult because the planner must ensure that the new linkings do not adversely affect other protected condition ranges in the GOST.

NONLIN can consider inserting new orderings into a plan by introducing or modifying links in two cases.

a) To detect and remove interactions: where one action interferes with an important effect of another and must be moved outside the effect's range.

b) To make a statement have a particular value at some node by linking another node which produces the required effect before the target node and ensuring the desired effect's range is protected accordingly.

NONLIN's strategy uses the GOST to determine the ranges of statements. Recall that the range of a GOST entry denotes the time between when a statement is given a particular value and the point at which it is required to maintain this value to satisfy the condition of some node. NONLIN ensures that there is no overlap between any ranges for which a statement P must have a value V and any ranges for which a P must have a value other than V. Two rules are used to maintain this consistency.

a) When there are multiple contributors to a condition on any node, the planner need only maintain at least one of them.

b) If the condition is only present to make a GOAL node a PHANTOM node then the planner can remove all the contributors. This will change the node to a GOAL node and force the planner to consider other ways to achieve the goal.

2.3.7 Summary

NONLIN still suffers from some major problems. The backtracking mechanism is inefficient. The macro-like nature of the task schemas is an example of shallow planning knowledge. The planner is limited in how it can reason about the underlying domain theory. For example, if an action contains a specific UNSUPERVISED condition that is not included in some higher-level action then the lower-level action cannot be applied because the planner cannot expand these types of conditions.

2.4 Other Single Agent Planners

2.4.1 ABSTRIPS

Sacerdoti developed ABSTRIPS (Abstraction-Based STRIPS) which modified the STRIPS search strategy to employ a hierarchical state-space search [Sacerdoti, 1974]. ABSTRIPS plans in a hierarchy of abstraction spaces in which successive levels of detail are introduced. It introduced the concept of criticality which was used to assign the preconditions of the domain operators to a specific abstraction level. The preconditions with the highest criticality value were deemed to be the most difficult to solve. ABSTRIPS used the goal stack planning strategy of STRIPS, but did so in hierarchical fashion. It would first try to solve the problem completely, considering only preconditions whose criticality value was the highest possible (i.e., preconditions with a lower criticality value were not placed on the goal stack). Once a complete solution was generated at this level of abstraction, it would use the constructed plan as an outline of the complete plan and consider preconditions at the next-lowest criticality level.

The technique employed by ABSTRIPS can greatly reduce the amount of search and backtracking that a planner must do. This is accomplished by focusing on important plan elements first and dealing with details only when a good plan outline has been produced. This allows the planner to eliminate alternatives very early and prune large subsets of the search in the process. Sacerdoti's research showed that ABSTRIPS spent

much less time than STRIPS in searching for solutions and visited far fewer nodes that were not on the correct path to a valid plan [Sacerdoti, 1974].

2.4.2 NOAH

NOAH (Nets of Action Hierarchies) was developed for use in assembly tasks. It introduced the concept of procedural nets [Sacerdoti, 1975]. A plan was represented in a procedural net at various levels of abstraction[6]. For example, NOAH could instruct a trained engineer to bolt the mounting bracket to the frame which is a high level instruction. We could also tell a novice how to accomplish this task in detail if necessary.

NOAH changed the search problem in planning to a space of partial plans instead of a space of world states [Sacerdoti, 1975]. For any non-primitive action in the network, the planner can consider any known method of reducing this action to a set of other actions or primitives. Planning in NOAH consists of choosing appropriate reductions from the sets of possibilities and ordering actions to eliminate harmful interactions. Each state in the problem space represents a set of possible plans. The more abstract the state, the more possible plans that could be generated. NOAH used a Hill Climbing search strategy to traverse the search space. When faced with a set of choices for reductions or orderings, NOAH uses local heuristics to pick one and discard the rest, never to consider them again. This means that some problems cannot be solved by NOAH.

2.5 Summary

Classical planners established the initial research in planning. Problems that can be solved by a single agent planner are limited by the agent's domain knowledge. Typically, a single agent can only solve relatively simple problems. In real world environments, problems are usually very complex. A complete, correct description of the world may not

[6]Procedural nets are also used as the basis for the planner's search space in NONLIN.

always be provided to the agent. Quite often the environment changes during plan execution or during planning. This results in great limitations in classical planners.

We need a planning system that involves different agents' expertise. Such a planning system should provide an environment to allow agents to contribute their domain specific knowledge to solve a complex problem. Sub-goals should be distributed among the agents and integrated together to form the final solution. Agents should also react to the environment when it changes instead of relying on the initial state of the world. When one agent fails to accomplish a task, the task should be sent to another agent with similar capability. Planning systems that address these objectives are discussed in the next chapter.

Chapter Three : Distributed Artificial Intelligence

3.1 Introduction

Most artificial intelligence (AI) research investigates how a single agent can exhibit intelligent behavior such as solving problems using heuristic or knowledge-based methods, planning, understanding and generating natural language, perception and learning [Hendler, Tate and Drummond, 1990]. Recent attempts to develop complex systems have revealed the shortcomings and problems of centralized, single-agent systems and have acted as a springboard for research in distributed artificial intelligence (DAI) [Durfee, Lesser and Corkill, 1989]. Several recent developments have together provoked interest in concurrency and distribution in AI: the development of powerful concurrent computers, the proliferation of multinode computer networks, and the recognition that much human problem solving and activity involve groups of people.

DAI can be divided into two primary arenas. Research in Distributed Problem Solving (DPS) considers how the work of solving a particular problem can be divided among a number of modules or *nodes* that cooperate at the level of dividing and sharing knowledge about the problem and about the developing solution [Bond and Gasser, 1988]. In a second arena, which we shall call Multiagent (MA) planning systems, research is concerned with coordinating intelligent behavior among a collection of (possibly pre-existing) autonomous intelligent *agents*, how they can coordinate their knowledge, goals, skills, and plans jointly to take action or to solve problems [Bond and Gasser, 1988]. To coordinate their actions, intelligent agents need to represent and reason about the knowledge, actions, and plans of other agents. DAI research can help us to improve our techniques for representing and using knowledge about beliefs, action, plans, goals, and so on. From a methodological perspective, virtually all research in DAI has focused on how a collection of agents can interact to solve a single common *global* problem.

3.2 Considerations in DAI

The basic questions that DAI must address are the following [Bond and Gasser, 1988]:

- How to formulate, describe, decompose, and allocate problems and synthesize results among a group of intelligent agents
- How to enable agents to communicate and interact: what communication languages or protocols to use, what and when to communicate, etc.
- How to ensure that agents act coherently in making decisions or taking action, accommodating the global effects of local decisions and avoiding harmful interactions
- How to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents in order to coordinate with them; how to reason about the state of the coordinated process
- How to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate their actions; how to synthesize views and results

3.2.1 Description, Decomposition, and Allocation of Tasks

Decomposition choices are critically dependent on how a problem is described, because it is the collection of attributes and descriptive categories for stating the problem that provides a language for expressing interproblem and interagent dependencies [Adler and Simoudis, 1990]. Formulation of problems requires some representation for the problem, as well as decisions on the boundaries of the problem and on what is known and unknown. Although there is little existing research in this area in DAI, some relevant research has occurred in knowledge-acquisition systems [Bond and Gasser, 1988].

The problem of task decomposition can be seen from several perspectives. In typical decomposition processes, a single *supertask* is decomposed into smaller subtasks, each of which requires less knowledge or fewer resources. The decomposition problem for multiple agents is more complex because of the need to match resources and capabilities

of different agents with appropriate tasks [Shaw, 1990]. Decomposition techniques need to account for the capabilities and resources of agents, and must make decisions about alternative types and granularity of decompositions. Intelligent approaches to task decomposition need to consider the representation of tasks, several dimensions of decomposition, available operators that can be applied to perform subtasks, available resources, and dependencies among tasks.

The problem of allocating particular tasks to particular agents is the problem of assigning responsibility for a particular activity. There are several choices of what aspects of a problem or task to provide in order to allocate responsibility for accomplishing that task to a particular agent. A task to be done can be completely or partially constructed by the agent itself, it can be assimilated via interaction with a *controlling* agent, or it can be given by a designer and embedded in the structure of the agent [Shaw, 1990]. Dynamic task allocation of any type requires reliable communication, coordination overhead, or redundancy.

3.2.2 Interaction, Language, and Communication

Interaction is important as a basic concept in DAI because it is the processes of interaction that make it possible for several intelligent agents to combine their efforts. Several dimensions of multiagent interaction are important for viewing organized aggregates. These include *among whom* the interaction takes place; *when* the interaction occurs; *what is the content* of the interaction or communication; *how the interaction is accomplished*; *why* the action occurs; and *what the basis of commonality is* [Huhns, Bridgeland and Arni, 1990].

When we work with distributed agents, we need to design and understand the language used for interaction, communication, and organization. This means that we need to know what knowledge to represent for communicating, and how to represent it in an interaction language. Communicating agents in general will have disparate knowledge, so

the language system may have to allow for differences in knowledge. Typical DAI systems have employed inflexible, predesigned communication languages [Huhns, Bridgeland and Arni, 1990]. For more adaptive DAI systems, we need more flexible approaches based on linguistic knowledge.

3.2.3 Coherence and Coordination

Coherence refers to how well the system behaves as a unit, along some dimension of evaluation [Durfee and Montgomery, 1990]. It includes the system's ability to reach satisfactory solutions, and the quality of the solutions it produces. The system's overall efficiency in achieving some end and the conceptual clarity of the system's actions are also included in examining the system's behavior.

Coordination, on the other hand, is a property of interaction among some set of agents performing some collective activity. Effective coordination implies some degree of mutual predictability and lack of conflict. The more unexpected conflict, the less well coordinated are the agents.

3.2.4 Modeling Other Agents and Organized Activity

Meaningful interaction between two agents requires that they have at least implicit knowledge of each other, such as the knowledge encoded in a communication protocol or language. Meaningful communication through language is impossible without some agreement on the intended effects of an utterance. One agent must know what reaction to expect on receipt of a message it sends, to plan communication intelligently. Coordination, which is important for avoiding harmful interactions and because local decisions have global effects, is possible only when some agent has some expectation about the character of the interaction. This expectation may be implicit, but it also may require reasoning.

3.2.5 Disparity and Conflict

Negotiation is a fundamental part of human cooperation, allowing people to resolve conflicts that could interfere with cooperative behavior [Bond and Gasser, 1988]. It is the process of improving agreement on common viewpoints or plans through the structured exchange of relevant information [Durfee, Lesser and Corkill, 1989]. Negotiation often is proposed in DAI research as a conflict-resolution and information-exchange scheme. Typically, negotiation involves some context, some sets of goals, some information or knowledge, and some procedure or protocol [Conry, Meyer and Lesser, 1986]. Negotiation has three important components: (a) there is a two-way exchange of information, (b) each party to the negotiation evaluates the information from its own perspective, and (c) final agreement is achieved by mutual selection [Davis and Smith, 1988]. Conflicting constraints can be bargained in the negotiation process. Agents may try to relax those constraints, or by reformulating a problem to eliminate them [Evans and Anderson, 1990].

3.3 Blackboard Systems

3.3.1 Introduction

In the mid 70's, the blackboard problem-solving model was developed [Engelmore and Morgan, 1988]. It formed the basis for early work in distributed artificial intelligence. The blackboard model consists of two basic components. The *knowledge sources* represent the knowledge needed to solve a problem. The knowledge is partitioned and kept separately and independently. The *blackboard data structure* represents a global database, the blackboard. Knowledge sources produce changes to the blackboard that lead incrementally to a solution to the problem. Communication and interaction among the knowledge sources take place solely through the blackboard. The objective of each knowledge source is to contribute information that will lead to a solution to the problem. Knowledge sources respond to the blackboard opportunistically. The blackboard uses a

centralized, agenda-based controller. The controller repeatedly polls the knowledge sources to see if they can contribute to the solution to the problem. The blackboard model is analogous to a group of experts trying to solve a complicated problem. Each has its own expert knowledge. A blackboard is placed in front of the group. A problem is described on the blackboard. All the experts monitor the blackboard continually. When they find areas that they can contribute to, they write their partial solutions to the blackboard to let other experts examine them. This process continues until a complete solution is found.

3.3.2 Framework

The general blackboard framework is shown in Figure 13 [Engelmore and Morgan, 1988]. The data on the blackboard are organized hierarchically. The knowledge sources are logically independent. Only the knowledge sources are allowed to make changes to the blackboard. On the basis of the latest changes to the information on the blackboard, a control module selects and executes the next knowledge source. Each knowledge source is responsible for knowing the conditions under which it can contribute to a solution. The knowledge sources respond opportunistically to changes on the blackboard. The control module monitors the changes on the blackboard and the contributions that the knowledge sources can make to decide what actions to take next. Various kinds of information are made globally available to the control module. The information can be on the blackboard or kept separately. The control information is used by the control module to determine the focus of attention at various times during the problem solving process.

Knowledge sources produce changes to the blackboard that lead incrementally to a solution, or a set of acceptable solutions, to the problem. The purpose of the blackboard is to hold computational and solution-state data needed by and produced by the knowledge sources. The knowledge sources use the blackboard data to interact with each other indirectly.

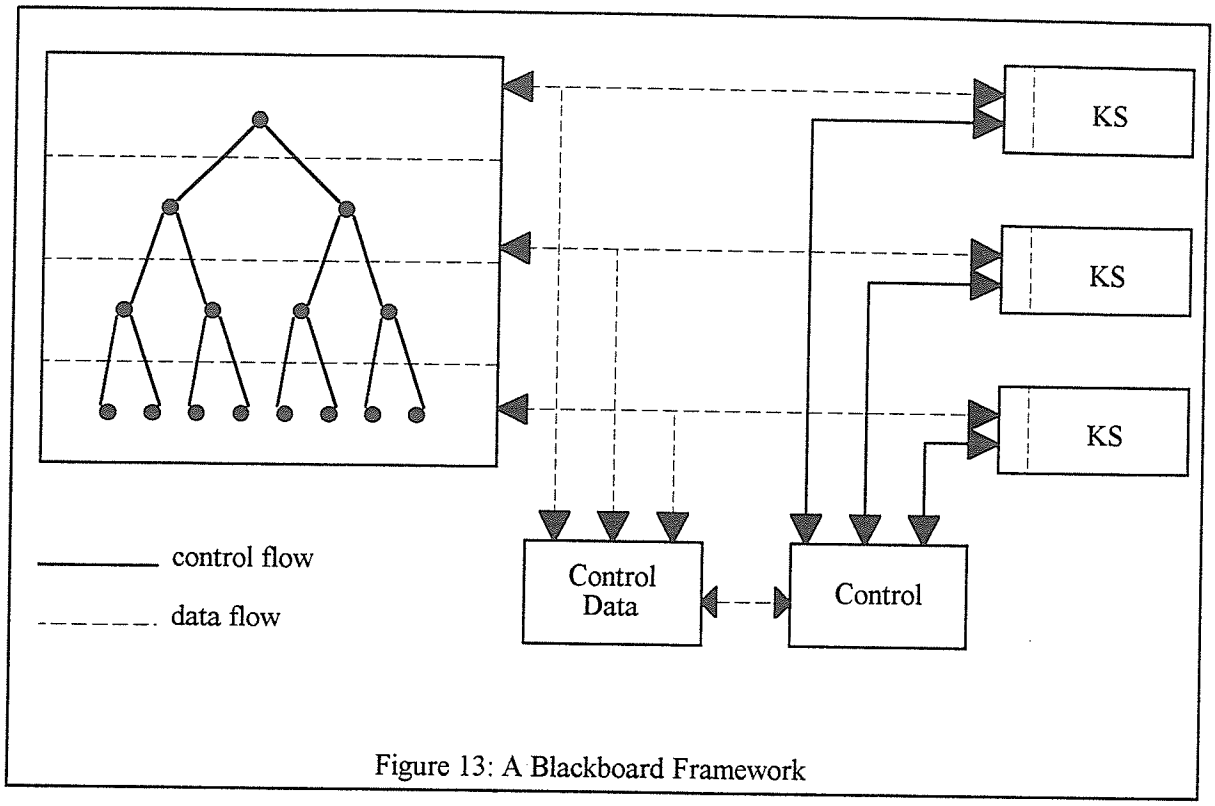


Figure 13: A Blackboard Framework

A Blackboard system is similar to a centralized multiagent planning system in that different knowledge sources contribute their expert knowledge to solve a problem. The knowledge sources in a Blackboard system, however, are not considered to be agents because they are typically not capable of functioning independently of the Blackboard system. An agent in a multiagent planning system usually can solve problems independently and in cooperation with other agents. Knowledge sources are similar to subroutines in conventional programming systems while agents are similar to complete programs in a multitasking environment.

The Blackboard model has been applied in DAI research to create individual agents that cooperate to solve problems [Hayes-Roth, 1988]. Each agent employs a local blackboard model which contains information about its local actions and those of some other agents in the system. We will introduce a multiagent planning model in Chapter 4

which utilizes the Blackboard model to implement a communication interface for individual agents.

3.4 Contract Networks

3.4.1 Introduction

A Contract network is a general purpose, task sharing model [Smith and Davis, 1988]. Nodes (or agents) coordinate their activities using a contracting protocol. By using negotiation based on task announcements, bids and awarded contracts, tasks are allocated to different agents. Manager nodes examine allocated tasks, decompose them, and broadcast task requests to other nodes. Interested nodes respond by making contract bids and become potential contractors. Interest is based on relevant expertise, resources, etc., of the node. Manager nodes review bids and select suitable contractors. This process may involve negotiation of bids to optimize the usage of resources. Contractors can become managers by decomposing their contracts into subcontracts. The Contract network model uses a mutual selection process rather than a manager-centered approach. It provides common message formats and negotiation protocols for interaction based on contracts, contract announcements, bid specifications, availability announcements and contract awarding [Davis and Smith, 1988].

3.4.2 Contract-Net Protocol

The nodes in a Contract-Net use a common communication protocol to form contracts concerning how they should allocate tasks in the network. Contracting involves an exchange of information between interested parties, an evaluation of the information by each member from its own perspective, and a final agreement by mutual selection. It differs from voting in that dissident members are free to exit the process rather than being bound by the decision of the majority.

In the Contract-Net protocol, nodes coordinate their activities through contracts to accomplish specific goals. Contracts are elaborated in a top-down manner. At each stage, a manager node decomposes its contracts into subcontracts to be accomplished by other contractor nodes. This process involves a bidding protocol based on a two-way transfer of information to establish the nature of the subcontracts and to determine which nodes will perform a particular subcontract [Smith, 1988]. The elaboration procedure continues until a node can complete a contract without assistance. The result of the contract elaboration process is a network of control relationships, in the form of manager and contractor relationships, distributed throughout the network [Smith, 1988].

Nodes allocate tasks in the following stages [Barr, Cohen and Feigenbaum, 1989]:

1. A manager forms a task to be allocated.
2. The manager announces the existence of the task.
3. Available nodes evaluate task announcement.
4. Suitable nodes submit bids for task.
5. The manager evaluates bids.
6. The manager awards contracts to the most appropriate node(s).
7. The manager and contractor communicate privately during contract execution.

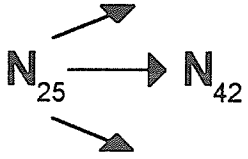
3.4.3 Example Application

A Contract-Net framework developed by Smith and Davis uses the model in a distributed interpretation application [Barr, Cohen and Feigenbaum, 1989]. The network tracks vehicles over a large geographical area. The spatially distributed network is composed of two types of nodes: sensor nodes that can extract signal features from the data they sense, and manager nodes that can process the signal features from several sensor nodes to construct a map of vehicle movements. A manager node wants to form contracts with sensor nodes that are adequately distributed around an area and that have a complement of sensory capabilities. On the other hand, a sensor node wants to interact

with nearby managers to minimize communication. The Contract-Net protocol allows manager and sensor nodes to each have input into the contracts that are formed.

This application illustrates the use of message structures in the Contract-Net protocol, depicted in Figure 14 [Barr, Cohen and Feigenbaum, 1989]. Every message includes information about its source, destination, type, and contract identifier. A task announcement message includes abstract information about the task, expected capabilities of potential contractors, the information that a bid should contain, and a deadline for when bids should be received. In the vehicle monitoring application, the task abstraction specifies the task type and the manager's location; the expected capabilities indicate that a contractor must have certain sensory abilities and be in a particular area; and the information a bid should contain includes the sensor's location and sensory abilities as shown in Figure 14 a).

Task Announcement



To: indicates a broadcast message.
From: 25
Type: TASK ANNOUNCEMENT
Contract: 22-3-1
Task Abstraction:
TASK TYPE SIGNAL
POSITION LAT 47N LONG 17E
Eligibility Specification:
MUST-HAVE SENSOR
MUST-HAVE POSITION AREA A
Bid Specification:
POSITION LAT LONG
EVERY SENSOR NAME TYPE
Expiration Time: 28 1730Z FEB 1979
Signal task announcement

Figure 14 a): Task Announcement

Task Bid

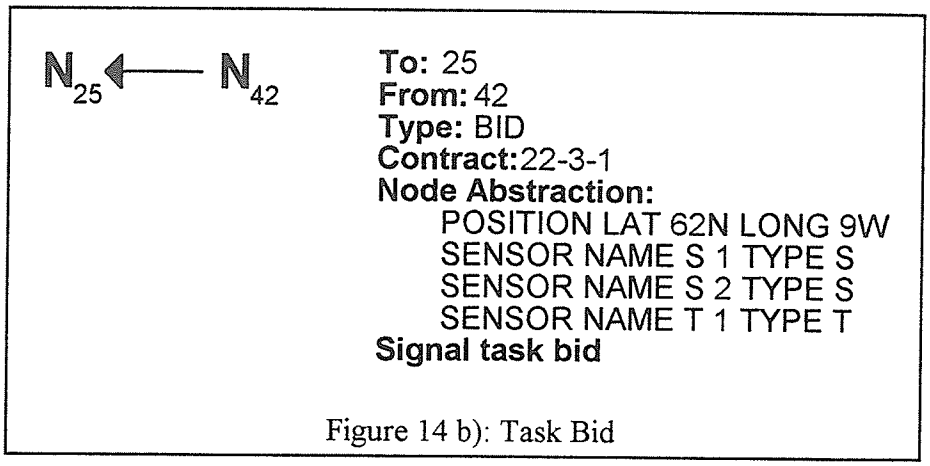


Figure 14 b): Task Bid

Task Award

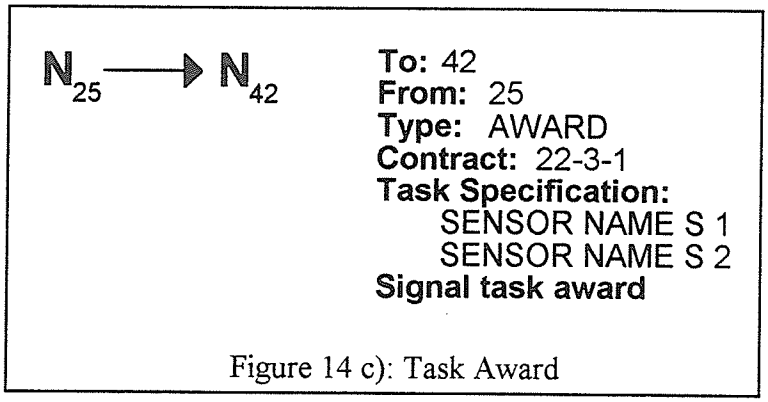


Figure 14 c): Task Award

Upon receipt of a task announcement, a node may send a task bid to the manager that announced the task. Besides the source, destination, type, and contract identifier, a task bid message includes the information requested in the task announcement's bid specification. In the vehicle monitoring application, the bid indicates the position and sensory capabilities of the sensor node as shown in Figure 14 b).

Finally, following the expiration of the task announcement, the manager evaluates the bids and builds a task award message for each node that is awarded the task. In the vehicle monitoring application, the task award message indicates which of a sensor node's sensory capabilities are requested by the manager as shown in Figure 14 c).

3.4.4 Contract-Net Summary

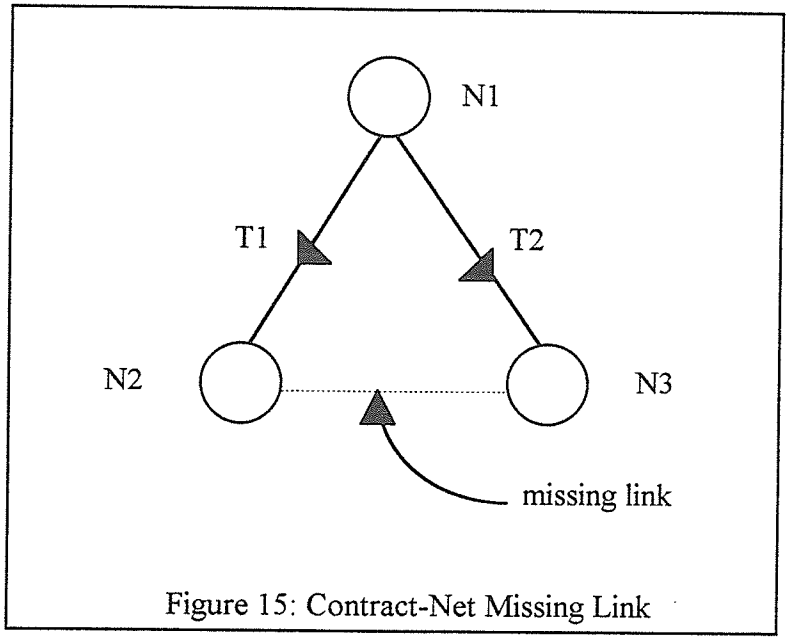


Figure 15: Contract-Net Missing Link

Since the contract network model relies heavily on interaction between nodes based on contract availability, bidding and awarding, it is susceptible to communication bottlenecks. In addition, contract networks provide no provisions for lateral communications between sub-tasks. As shown in Figure 15, node N1 sends task T1 to N2 and task T2 to N3. N1 forms a communication link with N2 and N3. There is no communication link created between N2 and N3. They may only communicate indirectly through N1.

The Contract network model concentrates on allocating tasks to increase parallelism and to make effective use of network resources. It assumes that the allocated tasks are independent, that is, that managers will decompose tasks to minimize subproblem interactions; and it assumes that the manager will implicitly know how to integrate the results of its contractors. In short, the contract net framework is geared toward top-down decomposition of large tasks and allocation of the subtasks. It is thus best suited for DAI

applications with well-defined task hierarchies, with tasks that are initially presented to a few nodes in the network and that can be decomposed into essentially independent subtasks.

3.5 Summary

There are many approaches for coordinating nodes in DAI, including contracting, negotiation, organizational structuring, multiagent planning, and sophisticated local control [Bond and Gasser, 1988]. From these very different approaches, we can infer that effective coordination requires three things. First, it requires structure because without structure the DAI nodes cannot interact in predictable ways. Structure is embodied in shared information such as organization and communication protocols. Second, effective coordination requires flexibility because DAI nodes typically exist in dynamically changing environments where each node might have incomplete, inaccurate, or obsolete information. Flexibility allows a contracting node to decide how to bid in its current situation, it allows a node in an organization to locally decide what partial solution to form given its current data, and it allows a node in a planning system to change its plan in response to changing circumstances. The third requirement for effective coordination is the knowledge and reasoning capabilities to intelligently use the structure and flexibility. Nodes must form and reason about what they are doing - their goals, plans, and beliefs - and how this fits into what they know about others. They must rely on structure to guide their activities to changing circumstances. In short, nodes need enough local sophistication to steer and appropriate course between regimentation and anarchy.

Chapter Four : Multiagent Planning

4.1 Introduction

Multiagent planning systems coordinate a number of agents to solve problems. Each agent contributes its expertise in a certain area so that a complicated problem can be solved by the group of agents. Usually such problems cannot be solved by an individual agent. Each agent in the system typically only needs a high level understanding of other agents. Such an approach naturally matches the way we do things in our daily lives.

In centralized multiagent planning, each agent forms its own local plan and distributes it to a central agent [Bond and Gasser, 1988]. The central agent reviews the plans submitted by other agents and analyzes them to identify critical regions. The plans may be modified and synchronization information may be inserted by the central agent when necessary. The central agent must have a complete picture of the entire goal to be achieved. This requires a significant amount of computation and communication resources.

In distributed multiagent planning, no single agent has a complete global view of the organization's activity. Agents plan together on a level-by-level basis and exchange partial plans based on models of other agents and their relevance [Bond and Gasser, 1988]. Agents examine other agents' partial plans and make adjustments in their own plans or make suggestions to other agents to change their plans.

We are going to describe the model developed originally by [Evans, 1988] and later extended by Evans and Anderson [Evans and Anderson 1989, Evans and Anderson 1990]. The description contained in this chapter are based largely on these references. This is a general purpose model and hence it is fairly complex. A prototype implementation of the model is presented in the next chapter.

4.2 Knowledge-based Model of an Agent

4.2.1 Basic Components

The basic components of an agent in our multiagent planning model are shown in Figure 16 [Evans and Anderson 1990]. Each agent is divided into two components: a problem-solving component that embodies the agent's problem-solving knowledge and skills; and a planning component that maintains and manipulates a knowledge-based model of the agent's own abilities and those of other agents in the environment to plan and coordinate cooperative problem-solving activities. The planning component acts as an intelligent coordination interface for the agent that determines when the agent should perform problem-solving functions for other agents and when the agent should have other agents perform problem-solving functions for it. The problem-solving component carries out those tasks that the planning component decides should be done by the agent itself. An agent's problem solving component is analogous to a single agent planning system. The agent's planning component enables the agent to plan and cooperate actions with other agents to solve problems that require the expertise and resources of several agents.

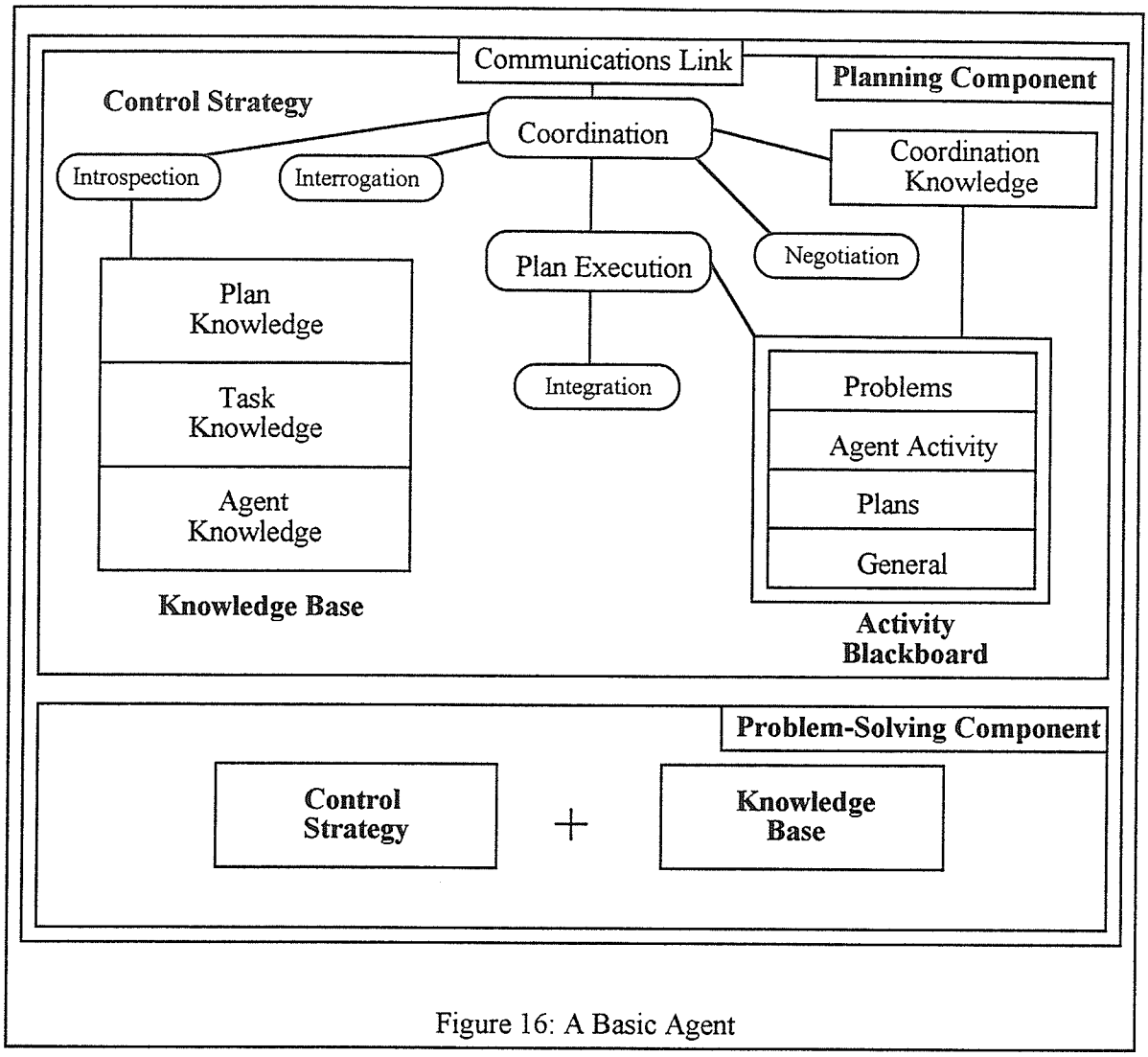


Figure 16: A Basic Agent

4.2.2 Constraints

Fox has done considerable work in analyzing the types of constraints that can be used in job-shop scheduling problems [Evans and Anderson, 1989]. Evans and Anderson have adapted this work to identify the types of constraints useful in planning and coordinating cooperative problem-solving activities [Evans and Anderson 1990]:

- Organizational Goals: maintaining an effective use of available resources by constraining their use. For example, preventing scarce resources (e.g. highly skilled agents) from being used on low priority tasks, preventing bottlenecks during cooperative work (deadlocks), and balancing the use of idle resources effectively.

- Physical Restrictions: identifying physical limitations of resources such as the processing time involved in performing particular tasks and the information that agents require to perform these tasks.

- Temporal Restrictions: identifying temporal orderings among the individual requests an agent receives and among the tasks required to carry out the requests.

- Availability Restrictions: identifying resource availability and the implications of unavailable resources (e.g. task X cannot be performed unless agent A is available).

- Preferences: identifying preferences for problem-solving methods and agents that can apply these methods.

Fox's work on the ISIS system led to the development of a constraint representation language capable of encoding a variety of knowledge about individual constraints including: expressions denoting the context in which a constraint is applicable; relaxation methods indicating how the constraint can be relaxed when conflicts arise; interactions among constraints that indicate interdependencies (i.e. satisfying one constraint may have a positive or negative effect on the ability to satisfy another); constraint generators that can introduce and propagate additional constraints dynamically when expectations are met or broken [Evans and Anderson, 1989]. In addition, constraints are assigned importance measures indicating the relative influence exerted by each constraint; some constraints must be satisfied, while others can be safely ignored or relaxed. Relaxation methods also are assigned measures indicating the relative utility and cost of each (i.e., preferences among available relaxation methods) [Evans and Anderson, 1990].

4.2.3 Communication Structures

Agents coordinate cooperative problem solving through the exchange of information with one another. We divide this information into two categories: *problem-solving requests* and *notifications* [Evans and Anderson 1990].

Problem-solving requests are structures that describe an operation (or set of operations) to be performed, an object (or set of objects) to which the operation is to be applied, and possibly an agent (or class of agents) that is to carry out the operation. A request is represented as a frame consisting of several slots. Each slot describes particular characteristics of the request. Each slot can have a variety of information associated with it, including: values, procedures, and constraints.

Notifications are more general structures that can be used to convey other types of required information between agents. These types of information include logistical information such as agent work load reports or notification of non-functioning agents. Besides the *housekeeping* information, notifications can convey control information such as global plan synopsis reports, descriptions of the current and future focus of attention, and most importantly constraint violation reports and relaxation notifications [Evans and Anderson, 1990].

4.2.4 Knowledge Sources

To coordinate cooperative problem-solving activities, a planning component requires extensive knowledge describing the types of requests the agent can process, the methods it can use, and the agents that can carry out the problem-solving specified by the various methods [Evans and Anderson 1990]. We divide the knowledge base into three distinct categories as illustrated in Figure 16. Each of these categories is represented as a collection of knowledge sources (KS) consisting of two components: a precondition component and an action component. The precondition is a request structure defining the form and content of the requests that the KS action can manipulate. Action components describe problem-solving methods, including information about the agents that can apply the methods and any additional constraints associated with the application of these methods.

The first category of knowledge, *plan knowledge*, specifies methods that can be used to process various composite requests an agent may receive. These are requests that can be transformed into a set of simpler requests that can be solved individually and then integrated to produce a final solution. Plan knowledge therefore represents *divide-and-conquer* methods.

The second category of knowledge, *task knowledge*, deals with what we refer to as *primitive requests*: that is, requests that can be distributed directly to an appropriate agent. The recipient of such requests may execute them immediately (i.e. pass the requests to its problem solving component) or the recipient may view the request as a composite request and transform it, distribute a set of simpler requests, and integrate the results [Evans, 1988].

The third category of knowledge, *agent knowledge*, describes specific agents and classes of agents residing in the environment, including communication protocols required to interact with agents and agent-specific preconditions that are imposed on various tasks the agents can perform.

In conjunction with the above mentioned categories of KSs, a planning component maintains a *coordination knowledge base* which is used by the control strategy to determine the amount of effort that is to be expended in processing requests and to focus attention towards the highest priority requests and the KSs best suited to each request. Much of this knowledge is represented as constraints which ensure that the agent expends its resources effectively and efficiently.

4.2.5 Activity Blackboard

The activity blackboard is a multi-partition working memory structure used to store information describing problem-solving requests being processed, plans designed to fulfill these requests, activities scheduled to carry out the plans, and general information about the current processing loads of the agent [Evans and Anderson 1990]. The most

important area of the blackboard is the plan partition, consisting of one or more *plan trees* which represent information about the problem-solving methods (KSs) applicable to each request currently being processed.

Plan tree nodes may be divided into two categories: AND nodes, representing required plan components (e.g. tasks specified in a plan KS); and OR nodes, representing one or more alternative plans, tasks, or agents. Each plan tree node is represented as a frame structure with several special slots specifying control information indicating how to interpret the information contained in a node, and a constraint list consisting of constraints associated with the node and any of its subordinates. The model employs a distributed multiagent planning approach so each agent's activity blackboard represents its view of the overall problem solving process (which may be inaccurate or in conflict with that of other agents).

4.3 Constraint-directed Planning

4.3.1 Planning Control Strategy

When requests are received, they are stored as entries in the problems partition of the activity blackboard (see Figure 16). Each request is assigned a priority based on coordination knowledge, and the highest ranked request is then selected and used to create a root node of a plan tree in the plan partition. There may be several plan trees in the plan partition, each corresponding to a request being processed. Only one plan tree will, however, be active at any given time; agents can perform only one task at a time, regardless of how much work is pending.

The control strategy begins to expand the current plan tree by selecting plan KSs that are applicable to the root node (request) [Evans and Anderson 1990]. Constraints are propagated from nodes at higher levels in the plan tree to ensure consistency is maintained. Each node in the plan tree is ranked and the control strategy will focus its

attention towards the highest ranked options whenever possible. Alternatives are explored when necessary.

A plan is considered executable when a complete path from the root node to a set of leaf nodes representing primitive tasks is generated. Once an executable plan is generated, the control strategy determines the task and agent KSs that can be used to distribute the primitive tasks to appropriate agents.

4.3.2 Plan Evaluation

To limit the number of applicable KSs, each applicable KS is assigned a utility measure indicating the relative utility of the method (i.e. ranking the ability to carry out the required actions); and a compatibility measure indicating the degree of compatibility between the activity demanded by the request and a KS precondition, ranging from 0 (completely incompatible) to 1 (completely compatible). Utility measures may be derived as a function of request characteristics, or may be predefined. An unsatisfied constraint reduces the compatibility measure by a domain-specific factor of the constraint's importance, while the successful relaxation of a constraint increases the compatibility measure [Evans and Anderson 1990].

4.3.3 Constraint Relaxation

Interactions among agents will often contain incompatibilities in the form of conflicting constraints. In many cases, however, incompatibilities can be overcome if the agents can negotiate with one anther to reformulate parts of a request or its corresponding plan tree when incompatibilities arise. Negotiation can be viewed as the process of relaxing constraints to propose alternative problem configurations that are satisfying and feasible to all those involved [Evans and Anderson 1990].

When negotiation is employed, the selection of applicable KSs becomes much more complicated - partially compatible KSs can potentially be used if relaxation methods

can be applied successfully to resolve incompatibilities. As a result, control requires a form of hypothesize-and-test search: when a KS is only partially compatible with a particular request, the control strategy must hypothesize reformulations based on the relaxation methods specified in the KS or in the request, and then test these hypotheses to ensure they are valid [Evans, 1988]. Validating reformulations involves propagating relaxations to ensure that other constraints imposed by the agents involved are not adversely effected.

The negotiation process is susceptible to a combinatorial explosion of relaxations that can potentially be applied to resolve conflicts. However, the negotiation process becomes much more selective when the utility and cost of available constraint relaxation methods are considered. In much the same manner as utility and compatibility measures are used to rate applicable KSs, utility measures and cost measures are associated with constraint relaxation methods to rate potential means of negotiation. For example, if another equally useful but less expensive relaxation method was found in a negotiation, it would be selected over the other relaxation methods. Each request is assigned two sets of thresholds: applicability thresholds and negotiation thresholds. Applicability thresholds are used to set minimum bounds for the utility and compatibility of plan alternatives, while negotiation thresholds set minimum bounds for the utility and maximum bounds for the cost of applicable constraint relaxation methods [Evans and Anderson, 1992].

4.4 Basic Functions

To plan interactions with other agents, there are some basic functions that an agent must be able to perform. The agent must be able to retrieve relevant plans and organizational knowledge [Evans and Anderson 1990]. The planning knowledge and organizational knowledge can be stored locally inside the agent itself as in distributed multiagent planning or stored in a global area as in centralized multiagent planning. Once the agent retrieves the plan, it must be able to interpret the plan and coordinate plan refinement. In case the agent needs more information, it can ask other agents to supply the

required information. While the plan executes, an agent must monitor the plan execution and react accordingly. When the environment changes, it should be able to replan the execution based on the new environment (possibly requiring interaction with other agents). This is very different from the state space approach that assumes the initial environment never changes. During the planning process, the agent must be able to coordinate the reformulation of plans to resolve conflicts between the plans created by other agents and its own plan. This may require negotiation with other agents to find the best solution. After the plans execute, agents in the system must be able to integrate the results of the plan steps together to form the complete goal.

4.4.1 Task Decomposition

In multiagent planning systems, agents need to decompose high-level tasks, assign sub-tasks among themselves, and combine the results of these sub-tasks. Task decomposition requires the agent to decide which agent does what task and when to do it [Evans and Anderson 1990]. In distributing the tasks among the agents, the tasks must be formulated and described appropriately. Tasks must be allocated to particular agents that will perform them or decompose them further and coordinate the execution and integration of the resulting tasks. The languages used for task description can greatly influence task decomposition, particularly the ability to perform automated task decomposition. Task description can influence how tasks can be decomposed and how they must be allocated. It is not unusual to have more than one way to decompose a task. The choices of decomposition can affect how tasks can be allocated because the skills of the agents allocated the tasks must eventually match the task requirements [Evans and Anderson, 1992]. Task decomposition is affected by dependencies among tasks that must be described at some level of the task descriptions. The dependencies may be physical, logical or temporal. Dependencies may also be statically or dynamically created. Task decomposition can be viewed as plan construction in multiagent planning that uses skeletal

plans of some form describing tasks and task dependencies. Task decomposition can be viewed as an AND-OR graph evolving from a single super-task. Alternative tasks and sub-tasks are decomposed to form OR-branches. Selection of decomposition is influenced by knowledge of available operators and agents capable of applying these operators. The selection must also consider temporal dependencies among tasks, resource conflicts and availability, and complimentary or mutually exclusive tasks. Problem descriptions and problem decompositions are distributed among agents and are represented at various levels of abstraction. No single agent needs to have a complete problem description or problem decomposition except in centralized multiagent planning or systems that share one common organizational representation among agents [Evans, 1988].

4.4.2. Interaction and Communication

Different agents in a multiagent planning system share their expertise by means of interaction and communication. Interaction occurs when one agent takes an action that has been influenced by the presence or knowledge of one or more other agents. Interaction can occur with or without explicit communication. Agents may communicate to establish terms and conditions of interactions or they may react based on models of each other. Models of other agents drive the initial construction of interactions and then agents communicate to confirm or reformulate interactions as necessary. By using models of other agents, the amount of communication that is necessary to coordinate interactions can be reduced. There are two types of interactions. By means of *routine interactions*, agents react to the influence of other agents without explicit confirmation of terms. On the other hand, *non-routine interactions* require agents to evolve the actual terms and details of interactions. Among large numbers of agents in a multiagent planning system, agents need to decide when to communicate and with whom. Agents can improve focus of communication by storing knowledge of relevant or potentially interested agents [Bond and Gasser, 1988].

To interact, agents must share a common language. Dialog among agents must be built from a common dictionary of terms that are used to construct and interpret messages. Agents must also be able to maintain common interpretations of message meanings even in the face of potentially disparate knowledge among agents. Agents may need to communicate their mutual beliefs, their knowledge of one another, their current goals and differences of opinions. For example, in contract nets, a common internode language was developed. It consisted of message types such as task announcements, bit messages and award messages. It used frame-like structures for each type of message. It also provided a context for building and interpreting messages.

4.5 Summary

We have described a knowledge-based model for employing constraint-directed reasoning in planning and coordinating cooperation among groups of problem solvers in a multiagent planning system. Cooperation is viewed as a multiagent constraint-satisfaction task in which agents interact to communicate constraints and work together to selectively relax constraints when conflicts occur. The model relies heavily on the assumption that the agents share common organizational goals and are willing to cooperate with one another to achieve those goals. The model is designed to be used in the creation of distributed expert systems to enable several knowledge-based agents to solve problems cooperatively. In the next chapter, we will examine how agents can cooperate with each other in a simulated environment that implements a simplified version of the model presented in this chapter.

Chapter Five : Implementation

The major focus of this thesis involves the design of control regimes and knowledge representations that allow agents to reason about local activities and cooperatively coordinate global activities with other agents. This chapter presents a detailed description of the implementation of a prototype of the multiagent model introduced in Chapter 4. The system is implemented using Macintosh Common LISP and runs on a single processor machine. The system can potentially be expanded to a multi-processor machine but this is beyond the scope of this thesis.

This chapter is divided into two parts. The first part describes the data structures used in the system. It presents detailed descriptions of each data structure defined in the model. The second part describes the program structure of the system. It presents the algorithms used in implementing the model.

5.1 Data Structures

To coordinate cooperative problem-solving activities, a planning component requires extensive knowledge describing the types of requests the agent can process, the methods it can use, and the agents that can carry out the problem-solving specified by the various methods [Evans and Anderson 1990]. The first four data structures (*plan structure, task structure*, *agent class structure* and *agent structure*) are used to represent individual instances of these knowledge sources. Data structures are also needed to encode requests and notifications distributed among agents during interagent communications. These data structures are presented in this section along with samples from the house building application.

Data structures in the system are represented as frames. Each frame structure acts as a template which can be instantiated with specific values. Default values can also be associated with various components of a frame.

5.1.1 Plan Structure

Plan knowledge specifies methods that can be used to process various composite requests an agent may receive. These are requests that can be transformed into a set of simpler requests that can be solved individually and then integrated to produce a final solution.

A plan is made up of:

plan-name that uniquely identifies the plan;

task-name that indicates when the plan is applicable (i.e., when it matches incoming tasks);

task-list that contains a list of tasks comprising the plan;

task-constraints that stores the constraints that the task may carry (note that constraints are not used in the current implementation of the model).

```
(defstruct PLAN
   plan-name
   task-name
   task-list
   task-constraints)[6]
```

An instance of a plan defined in an agent of the house building application can be created as follows:

```
(MAKE-PLAN :plan-name 'Plan-Mark-House-Building
           :task-name 'House-Building
           :task-list '(Build-Exterior Build-Interior)
)
```

The name of the plan is *Plan-Mark-House-Building*. The name of the incoming task that it can handle is *House-Building*. Tasks result from transforming *House-Building* into simpler tasks : *Build-Exterior* and *Build-Interior*. In this case, *task-constraints* is not instantiated with any value. The default value for *task-constraints* is *nil*.

---

[6] *defstruct* creates a data structure template and returns its name. Any number of components with default values can be used. As a side effect, a constructor function to be referenced as *make-<structname>* is created that can be used with *setf* to define specific objects that have this structure []

5.1.2 Task Structure

Task knowledge deals with what we refer to as *primitive requests*: that is, requests that can be distributed directly to an appropriate agent. The recipient of such requests may execute them immediately (i.e. pass the requests to its problem solving component) or the recipient may view the request as a composite request and transform it, distribute a set of simpler requests, and integrate the results.

A task is made up of:

task-name that uniquely identifies a task;

agent-class-list that contains a list of classes of agents that are known to be capable of performing the task;

task-constraints that contains a list of constraints included to constrain the form of a task that the list of agent classes can solve.

```
(defstruct TASK
   task-name
   agent-class-list
   task-constraints)
```

An instance of a task defined in an agent of the house building application can be created as follows:

```
(MAKE-TASK :task-name 'Build-Exterior
           :agent-class-list '(Mark-Class-Build-Exterior)
)
```

The name of the task is *Build-Exterior*. The agent classes listed are used to find specific agent instances that can potentially perform the task. The *agent-class-list* contains only one agent class which is *Mark-Class-Build-Exterior*. In this example, *task-constraints* is not instantiated with any value. The default value for *task-constraints* is *nil*.

5.1.3 Agent Class Structure

Agent class knowledge specifies individual agents known to belong to a specific agent class. It is used in conjunction with the task knowledge to select one or more agents to whom a given task can be sent for processing.

An agent class is made up of:

class-name that uniquely identifies an agent class;

agent-list that contains a list of agents belonging to the class. Note that the agent list may differ for each agent representing varying views of the group.

```
(defstruct AGENT-CLASS
   class-name
   agent-list)
```

An instance of an agent class defined in an agent of the house building application can be created as follows:

```
(MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Exterior
                  :agent-list '(John-House-Building)
)
```

In this example, the agent class name is *Mark-Class-Build-Exterior*. There is only one known agent that belongs to this class - *John-House-Building*.

5.1.4 Agent Structure

Agent knowledge describes specific agents and classes of agents residing in the environment, including communication and activity structures. The simulator has predefined information about each agent - this information is represented as instances of the *agent structure*. The information forms the knowledge base of the individual agents. The simulator uses this information to simulate each agent's actions in response to requests and notifications that are distributed among the agents.

An agent is made up of:

agent-name that uniquely identifies the agent;

plan-list that contains a list of plans (plan structures) that the agent can carry out to solve tasks;

task-list that contains a list tasks (task structures) that can be carried out directly by the agent without being broken down using a plan;

agent-class-list that contains a list of agent classes (agent class structures) known to the agent;

constraint-list that contains a list of constraints applicable to the agent's actions;

incoming-request-list that contains a list of requests the agent has received from others;

outgoing-request-list that contains a list of requests the agent has sent to others;

incoming-notification-list that contains a list of notifications received from other agents;

outgoing-notification-list that contains a list of notifications sent to other agents;

activity-blackboard that describes the agent's current and planned activities;

processing-constraints that defines the constraints imposed on the agent for processing requests and notifications on each simulation cycle[7].

```
(defstruct AGENT
   agent-name
   plan-list
   task-list
   agent-class-list
   constraint-list
   incoming-request-list
   outgoing-request-list
   incoming-notification-list
   outgoing-notification-list
   activity-blackboard
   processing-constraints)
```

[7]These constraints enable the user to examine the effects of changing the workload capabilities of individual agents.

An instance of an agent defined in the house building application can be created as follows:

```
(setq agent-1 (MAKE-AGENT :agent-name 'Mark-House-Building
                  :plan-list (LIST
                              (MAKE-PLAN :plan-name 'Plan-Mark-House-Building
                                         :task-name 'House-Building
                                         :task-list '(Build-Exterior Build-Interior))
                              )
                  :task-list (LIST
                              (MAKE-TASK :task-name 'Build-Exterior
                                         :agent-class-list '(Mark-Class-Build-Exterior))
                              (MAKE-TASK :task-name 'Build-Interior
                                         :agent-class-list '(Mark-Class-Build-Interior))
                              )
                  :agent-class-list
                              (LIST
                               (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Exterior
                                                 :agent-list '(John-House-Building))
                               (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Interior
                                                 :agent-list '(John-House-Building)))
                  )
)
```

In this example, the name of the agent is *Mark-House-Building*. Since the *incoming-request-list, outgoing-request-list, incoming-notification-list, outgoing-notification-list* and *activity-blackboard* are initially empty, they are defaulted to have the value *nil*. When a simulation is executing, these fields are dynamically modified by the system based on the activities the agent becomes involved in.

5.1.5 Request Structure

Problem-solving requests are structures that describe an operation (or set of operations) to be performed, an object (or set of objects) to which the operation is to be applied, and possibly an agent (or class of agents) that is to carry out the operation. A request is represented as a frame structure consisting of several slots. Each slot describes particular characteristics of the request. Each slot can have a variety of information associated with it, including: values, procedures, and constraints.

A request is made up of:

request-id that uniquely identifies the request;

from-agent that identifies the agent which sent the request;

to-agent that identifies the agent that is to receive the request;

request-status that indicates the current status of the request;

request-type that identifies the type of request;

task-name that identifies the name of the task if the *request-type* is a task;

activity-node-id that identifies the node which is created for the request in the activity blackboard of the recipient of the request;

request-args that identifies the arguments associated with the request.

```
(defstruct REQUEST
   (request-id (gensym[8]))
   from-agent
   to-agent
   request-status
   request-type
   task-name
   activity-node-id
   request-args)
```

Requests are linked to entries on an agent's activity blackboard. An incoming request is linked to a plan that can be used to satisfy it or a primitive task when a plan is not needed. Outgoing requests are linked to tasks from one or more plans. These requests are sent to other agents to carry out part of the processing needed to satisfy some previously received tasks.

[8]Gensym is a builtin Common Lisp function that generates a unique symbol.

An example of a request used in the house building application can be created as follows:

```
(MAKE-REQUEST :from-agent 'Mark-House-Building
              :to-agent   'John-House-Building
              :request-type 'task
              :task-name  'House-Building
)
```

5.1.6 Notification Structure

Notifications are general structures that can be used to convey other types of required information between agents. These types of information include logistical information such as agent work load reports or notification of non-functioning agents. Besides this *housekeeping* information, notifications can convey control information such as global plan synopsis reports, descriptions of the current and future focus of attention, and constraint violation reports and relaxation notifications[9].

A notification is made up of:

notif-id that uniquely identifies a notification;

from-agent that identifies the agent which sends out the notification;

to-agent that identifies the agent which receives the notification;

notif-status that indicates the current status of the notification;

notif-type that identifies the type of notification;

request-id that identifies the request if the notification is sent as a result of processing associated with a specific request;

notif-info that contains the actual details of the notification.

[9]This feature has not been implemented in the prototype.

```
(defstruct NOTIFICATION
   (notif-id (gensym))
   from-agent
   to-agent
   notif-status
   notif-type
   request-id
   notif-info)
```

Notifications are usually linked to one or more requests that were sent out previously. They can provide results for requested tasks, constraint violation reports, negotiation information, or general information about an agent's status. The receipt of a notification is used to drive further processing of an agent's activity blackboard.

In the following example, a notification is sent from one agent to another agent indicating that a task is completed without any problem. After the agent receives the notification, it sets the activity status of the node in the activity blackboard with the matching request id to the value contained in the *notif-info* field. For more details, see section 5.2.4.

```
(MAKE-NOTIFICATION :from-agent   'John-House-Building
                   :to-agent     'Mark-House-Building
                   :notif-type   'task
                   :request-id   'E1234
                   :notif-info   'No-Problem
)
```

5.1.7 Activity Structure

The activity blackboard is a multi-partition working memory structure used to store information describing problem-solving requests being processed, plans designed to fulfill these requests, activities scheduled to carry out the plans, and general information about the current processing loads of the agent [Evans and Anderson 1990]. The most important area of the blackboard is the plan partition, consisting of one or more *plan trees* which represent information about the problem-solving methods (KSs) applicable to each request currently being processed. An activity frame represents a node in a *plan tree*.

An activity frame is made up of:

activity-id that uniquely identifies an activity;

task that represents the task associated with the activity node;

task-args that contains the arguments that comes with the task;

status that indicates the current status of the processing associated with the task;

cycle that indicates the current cycle that the agent is working on the task;

activity-type that identifies the type of activity node;

parent-task-id that contains the id of the parent task;

parent-request-id that contains the id of the request from which the task evolved;

applicable-agent-classes that contains the agent classes that a task can be distributed to;

current-agent-class that contains the current agent class selected to process the task;

applicable-agents that contains a list of agents that could be sent the task;

current-agent that contains the current agent selected to receive the task;

applicable-plans-and-tasks that contains a list of plans that can be used to solve the task associated with the activity;

current-plan-or-task that contains the current plan or task selected;

current-task-list that contains the details of the current plan.

```
(defstruct ACTIVITY
   (activity-id (gensym))
   task
   task-args
   status
   cycle
   activity-type
   parent-task-id
   parent-request-id
   applicable-agent-classes
   current-agent-class
   applicable-agents
   current-agent
   applicable-plans-and-tasks
   current-plan-or-task
   current-task-list)
```

Plan tree nodes may be divided into two categories: AND nodes, representing required plan components (e.g. tasks specified in a plan KS); and OR nodes, representing one or more alternative plans, tasks, agent classes or agents. Each plan tree node is represented as a frame structure with several special slots specifying control information indicating how to interpret the information contained in a node, and a constraint list consisting of constraints associated with the node and any of its subordinates. The activity frames represent the current activity associated with each request received by an agent. Processing associated with a global problem is distributed across several activity frames in one or more agents.

There are four types of nodes in a plan tree. Plan nodes represent all plans that can be used in fulfilling a request. Task nodes represent all tasks specified in a plan. Agent class nodes represent all agent classes of a task. Agent nodes represent all agents of an agent class.

Applicable-plans-and-tasks represents the plan nodes that can be used to solve the task associated with the activity. *Current-plan-or-task* represents a plan node that contains the current plan or task. *Task* represents a task node that represents the task associated with the activity node. *Applicable-agent-classes* represents the agent class nodes that a task can be distributed to. *Current-agent-class* represents an agent class node that

contains the current agent class selected to process the task. *Applicable-agents* represents the agent nodes that could be sent the task. *Current-agent* represent an agent node that contains the current agent selected to receive the task.

5.2 Program Structure

The prototype system presented in this chapter implements a general purpose simulator which manipulates the agent structures encoded in the system to simulate multiagent processing of requests and notifications. The basic algorithms used are described in detail in this section. Sample applications of the system are presented in the next chapter.

The structure chart shown in Figure 17 a) and b) describes the hierarchical structure of the system. The program code in the system can be found in Appendix B.

5.2.1 Basic Algorithm

Based on the structure chart shown in Figure 17 a) and b), the simulator examines every agent in the system on a round robin basis. A simulator cycle consists of applying the following processing to each agent as shown in Figure 18:

Process-Agents:

- Set up a list of agents to be processed by the system.
- Select an agent by calling the *Find-Agent* routine.
- Process the selected agent by calling the *Process-An-Agent* routine.

Process-An-Agent:

- Invoke the *Process-Requests* routine to process 0 or more requests pending processing in the incoming request queue.
- Invoke the *Process-Activity-Nodes* routine to process 0 or more activity nodes in the activity blackboard.

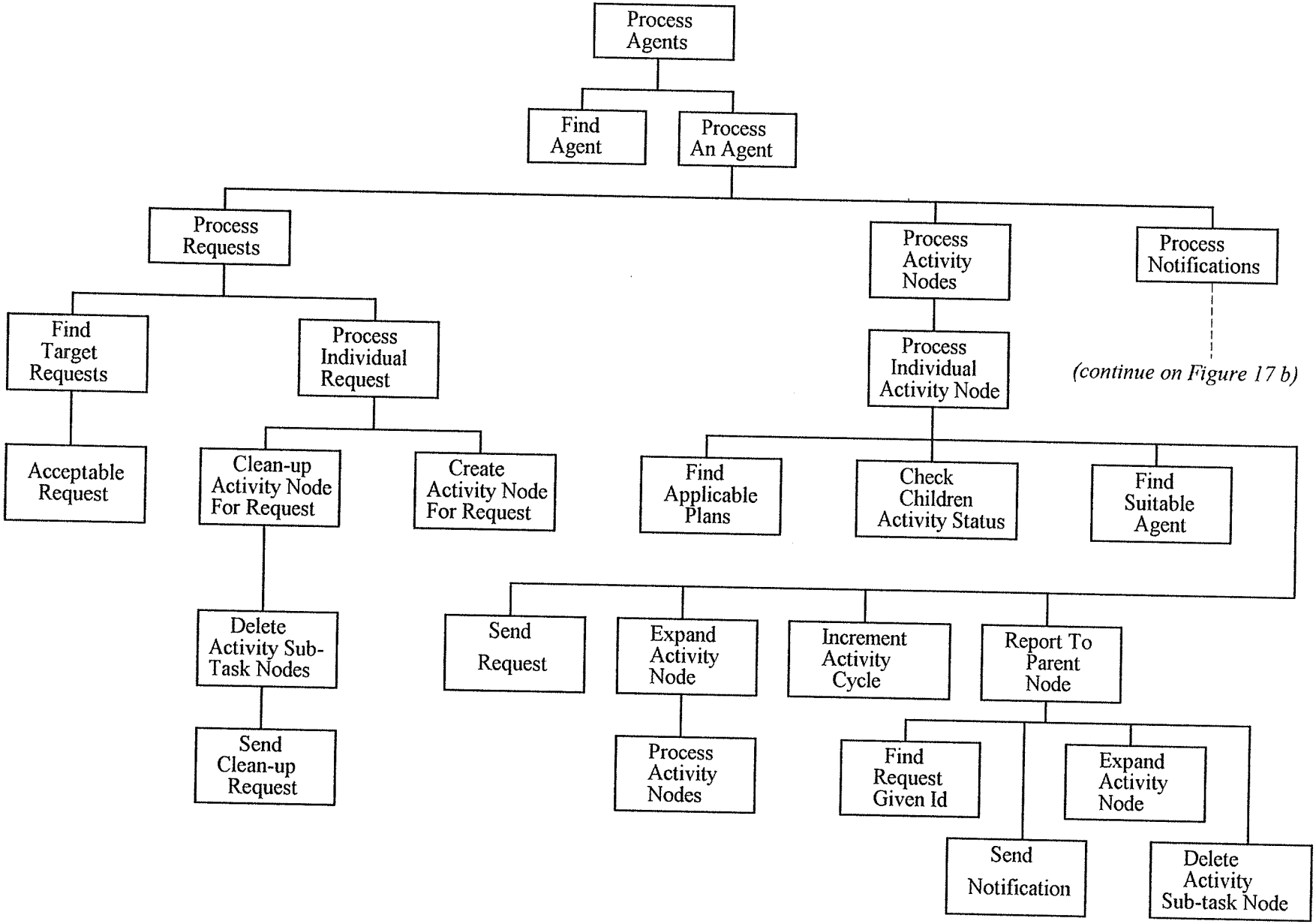


Figure 17 a): Program Structure Chart (a)

(continue from Figure 17 a)

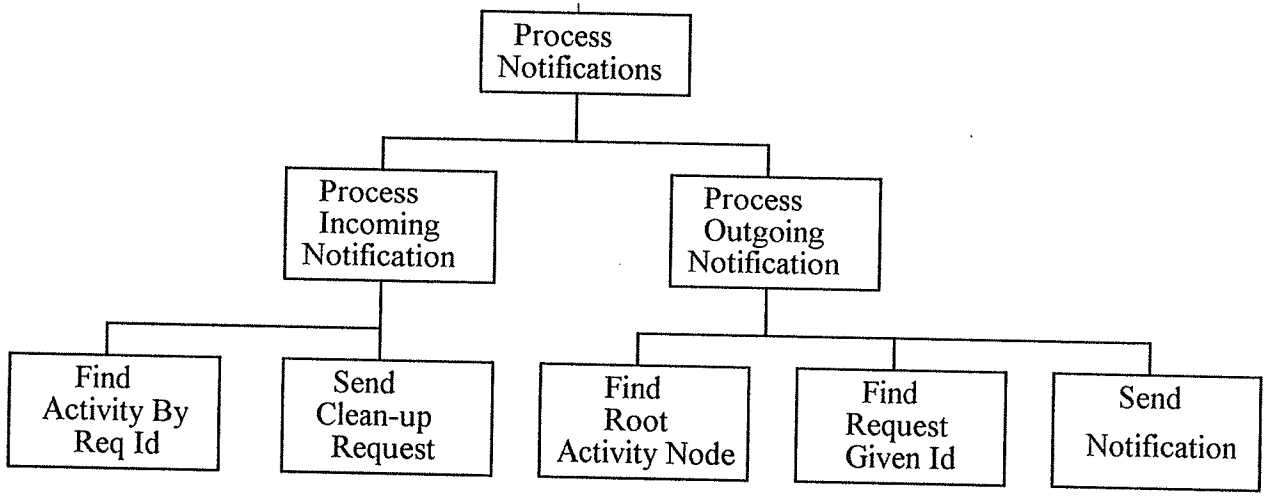


Figure 17 b): Program Structure Chart (b)

- Invoke the *Process-Notifications* routine to process 0 or more incoming notifications pending in the incoming notification queue.

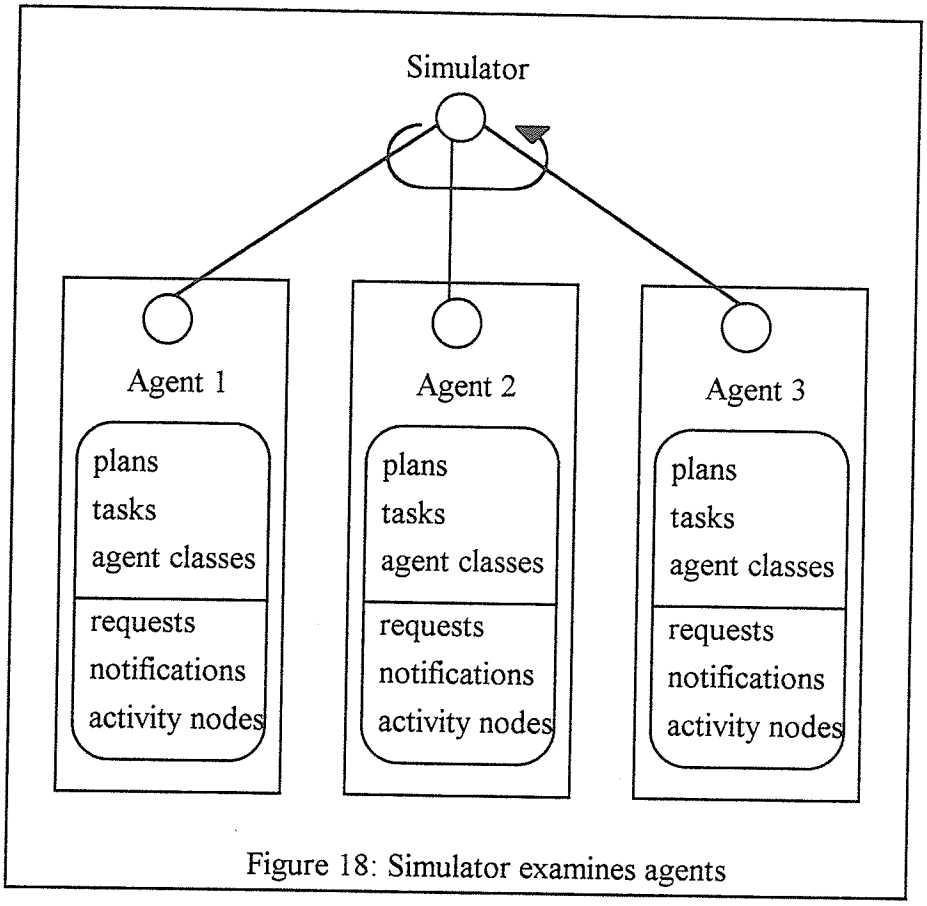


Figure 18: Simulator examines agents

5.2.2 Process Requests

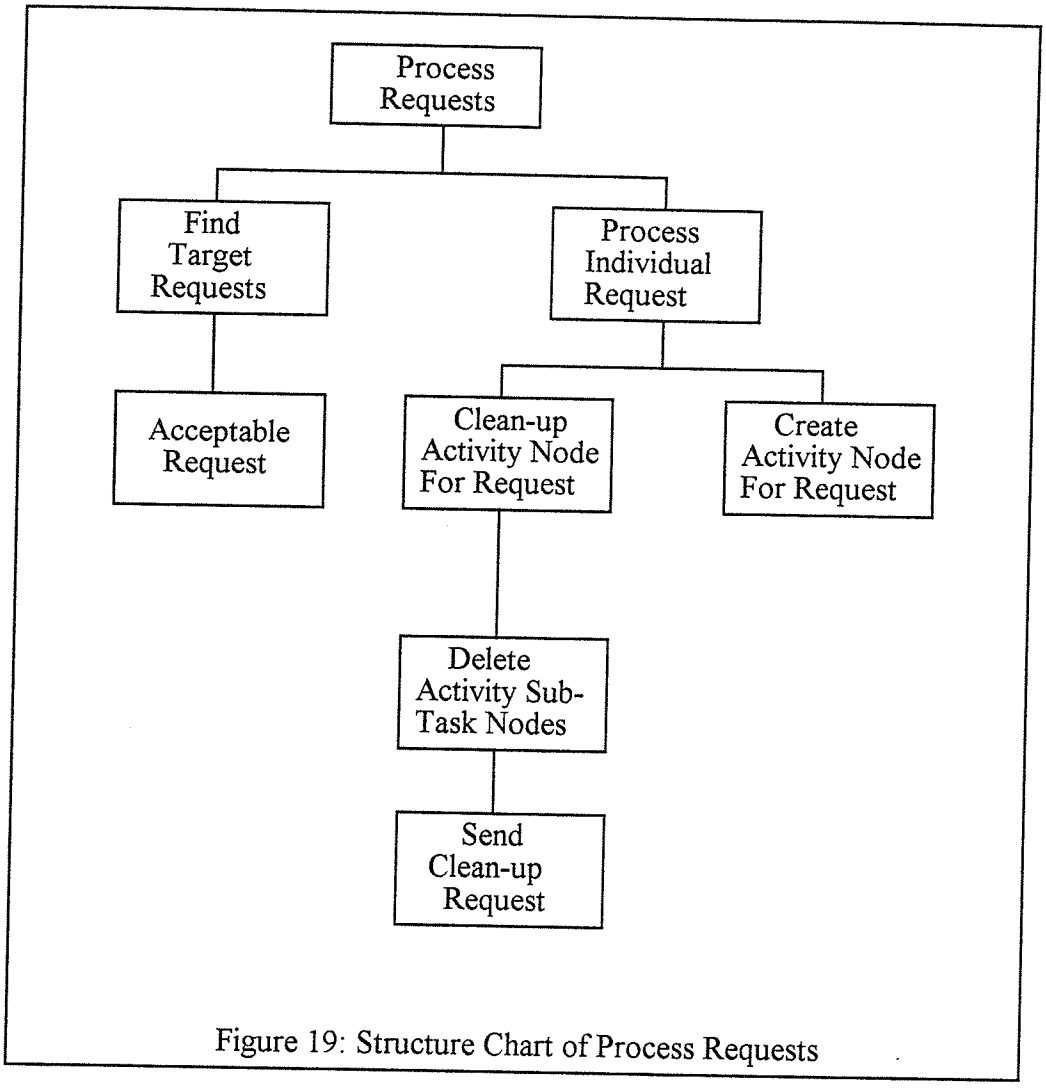


Figure 19: Structure Chart of Process Requests

The simulator performs the steps in Figure 19 to process requests for a specific agent. *Process-Requests* invokes *Find-Target-Requests* to build a list of request ids from the agent's incoming request queue. The *Acceptable-Request* routine is invoked from *Find-Target-Requests* to filter out unacceptable requests based on request criteria. *Process-Requests* then invokes *Process-Individual-Request* to process each of the remaining requests.

In *Process-Individual-Request*, if the incoming request is a *clean-up* request, *Clean-Up-Activity-Node-For-Request* will be invoked. *Clean-Up-Activity-Node-For-Request* invokes *Delete-Activity-Sub-Task-Nodes* to update the agent's activity blackboard by deleting any task nodes that have the given activity node as its parent. If the sub-tasks of the activity node have been sent to other agents, *Send-Request* will be invoked to send *clean-up* requests to those agents. On the other hand, if the incoming request is a request for new task, the routine will create a new activity node (root node) for the request by invoking *Create-Activity-Node-For-Request*.

5.2.3 Process Activity Nodes

Process-activity-nodes examines the activity nodes in the activity blackboard of a given agent. *Process-individual-activity-node* is executed for each activity node. If the activity node requires a new plan, *find-applicable-plans* will be executed to find plans that apply to the node. Figure 20 shows the structure chart of *Process-activity-nodes*. If a new plan is found for the node, the node will be expanded by *expand-activity-node*. Subordinate nodes will be created with the parent node id set to the current node id.

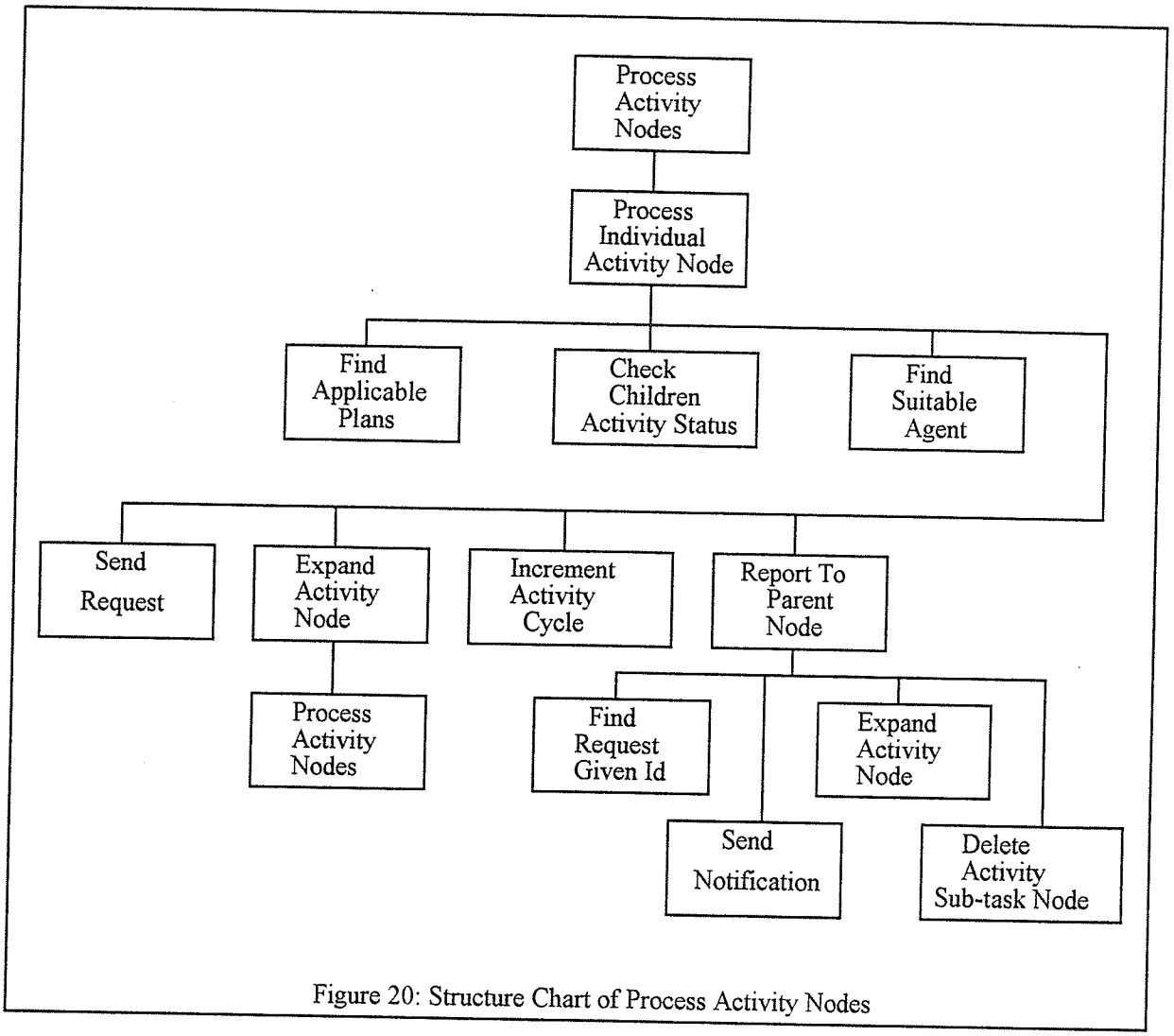


Figure 20: Structure Chart of Process Activity Nodes

The status of the subordinate activity nodes are checked by *check-children-activity-status*. If all the subordinate nodes have been completed successfully, the current node's status will be set to *done*. This in effect simulates the process of integrating the results of distributed tasks. If the status of the activity node is set to *done*, a notification will be sent to the requesting agent by executing *report-to-parent-node*. It finds the original incoming request (*find-request-given-id*) and sends the notification by calling *send-notification*.

If the agent must send a task to another agent, suitable agents will be found using the *find-suitable-agent* routine and a request will be sent to one of these agents by executing *send-request*.

If, after the agent tries all the possible agent classes and agents of a task in a plan and no agent can complete the task, the status of the task's associated activity node will be set to *problem*. In this case, the agent will delete the node's subordinate nodes by executing *delete-activity-sub-task-node* and will then try to find an alternative plan. If an alternative plan is available, it will expand the node using the new plan by executing *expand-activity-node*.

If the task is a primitive task (i.e., the task can be done by the agent itself), then after the activity node is processed once, the activity cycle parameter is incremented by one. The activity cycle is then compared with the task cycle constraint set in the task-constraints of the task. If they are equal, the activity status of the current node will be set to be *done*. The activity cycle gives the user the flexibility to vary the time required for a task to be completed. While one agent is performing one task in a plan, another agent may already be finished processing another task in the same plan.

All the activity nodes are created in the activity blackboard of the agent. Figure 21 a) and b) shows a blackboard containing more than one activity node. Each root node represents an incoming request and the subordinate nodes of the root node represent the subtasks after being broken down using a plan.

When there are two new incoming requests, two new root nodes are created in the activity blackboard as shown in Figure 21 a). Suppose a plan was found for the first node. This root node is then expanded to create sub-ordinate nodes based on the plan as shown in Figure 21 b). Suppose the task associated with the second root node can be done by the agent itself (i.e., it is a primitive task). The second root node will not be expanded.

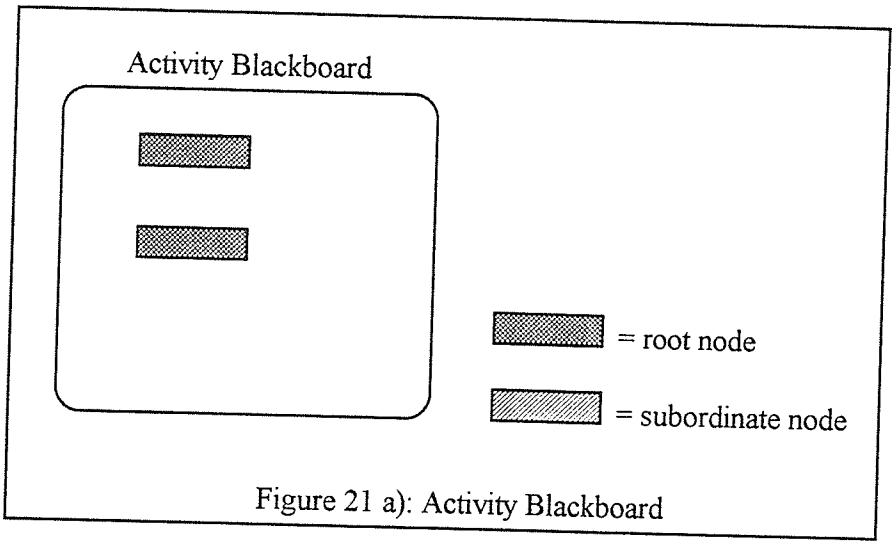


Figure 21 a): Activity Blackboard

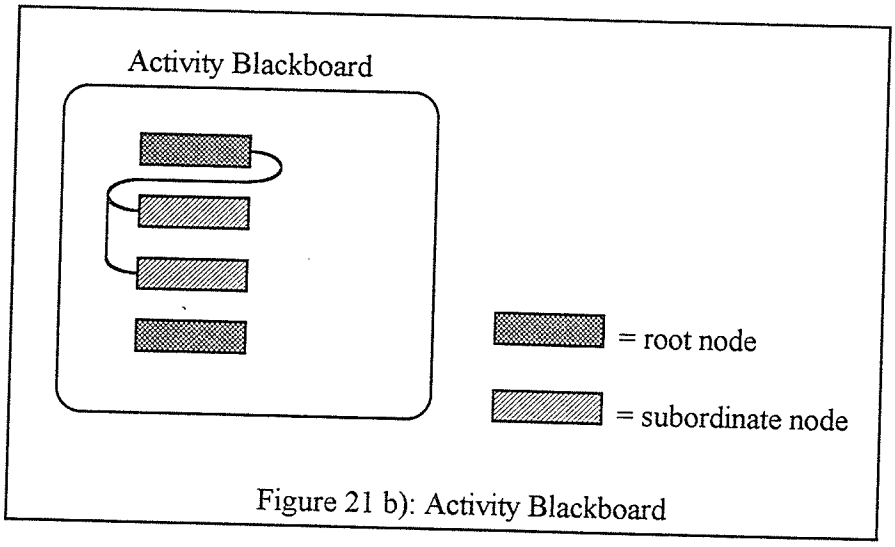


Figure 21 b): Activity Blackboard

Figure 22 a), b), c) and d) show the internal planning structure of an activity node. Figure 22 a) shows a set of applicable plans and tasks, and the currently chosen plan. It also shows all the applicable agent classes and the currently chosen agent class associated with each task in the chosen plan. Finally, it shows all the applicable agents and the currently chosen agent for each task.

When a currently chosen agent fails, an alternative agent can be chosen as shown in Figure 22 b). The newly chosen agent belongs to the same agent class as the previously chosen agent.

When all the agents of the same agent class fail, an alternative agent class can be chosen as shown in Figure 22 c). A new agent will be chosen from the new agent class. The newly chosen agent class is associated with the same task as the previously chosen agent class.

When all the agent classes associated with a task in a plan fail[10], an alternative plan can be chosen as shown in Figure 22 d). When this plan is expanded, a new set of tasks and agent classes results.

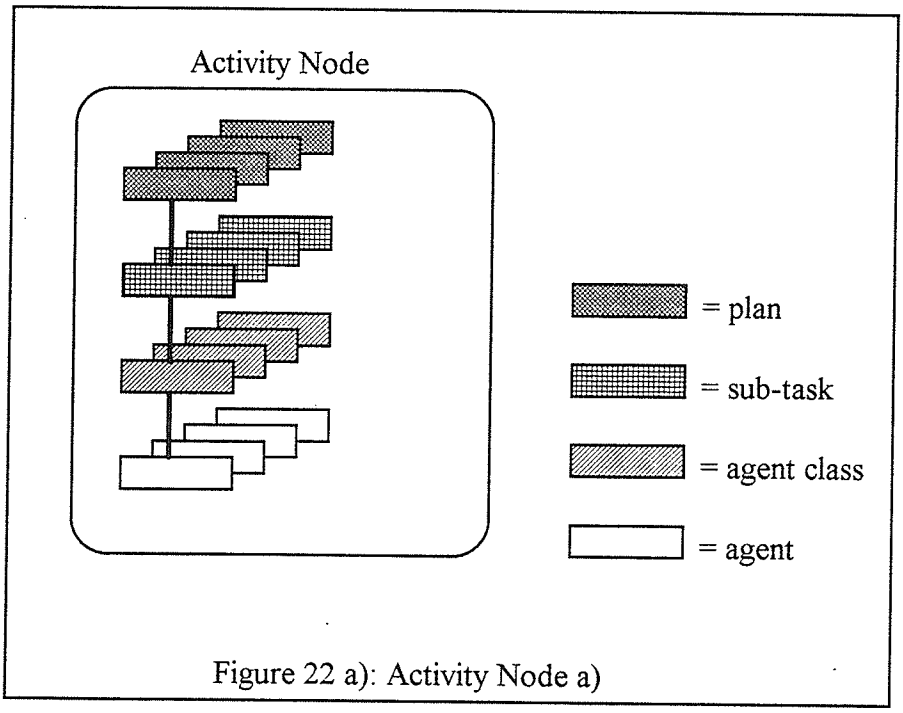


Figure 22 a): Activity Node a)

[10]Constraints can be associated with requests to limit the type and number of alternatives that should be explored when difficulties arise.

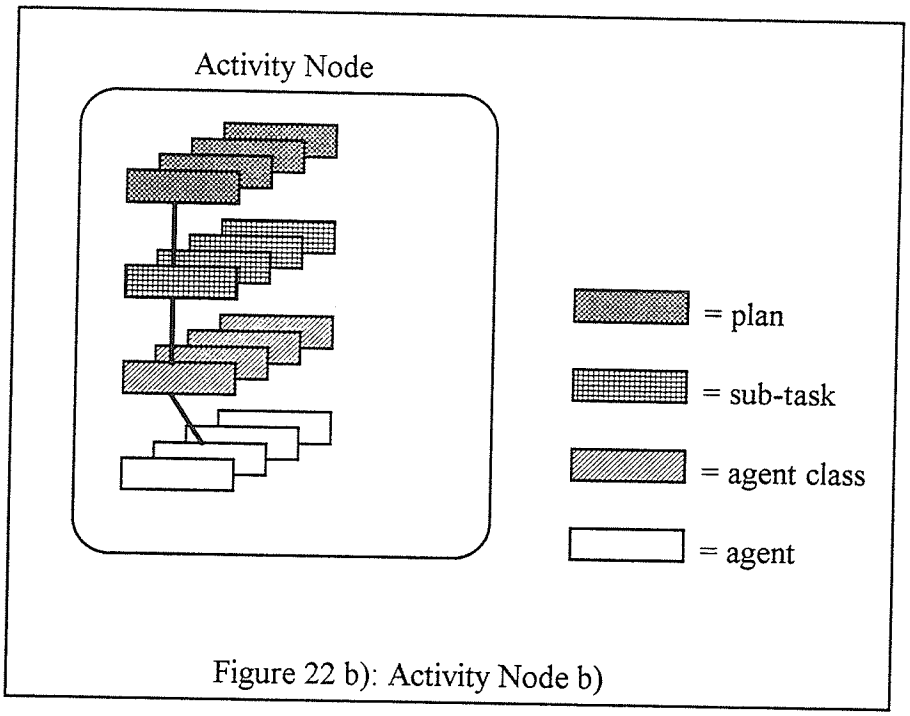


Figure 22 b): Activity Node b)

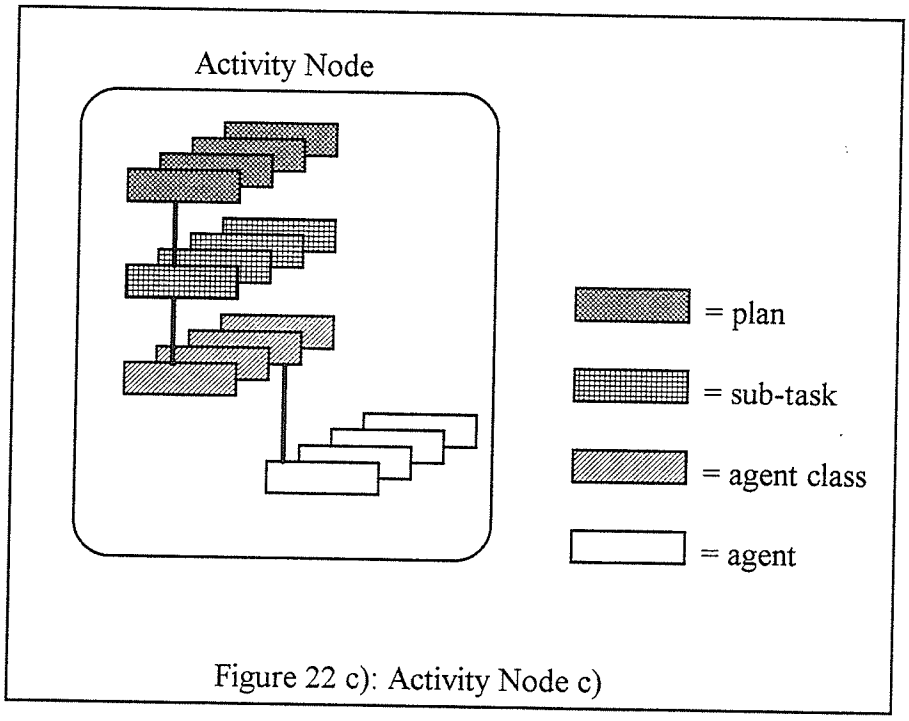


Figure 22 c): Activity Node c)

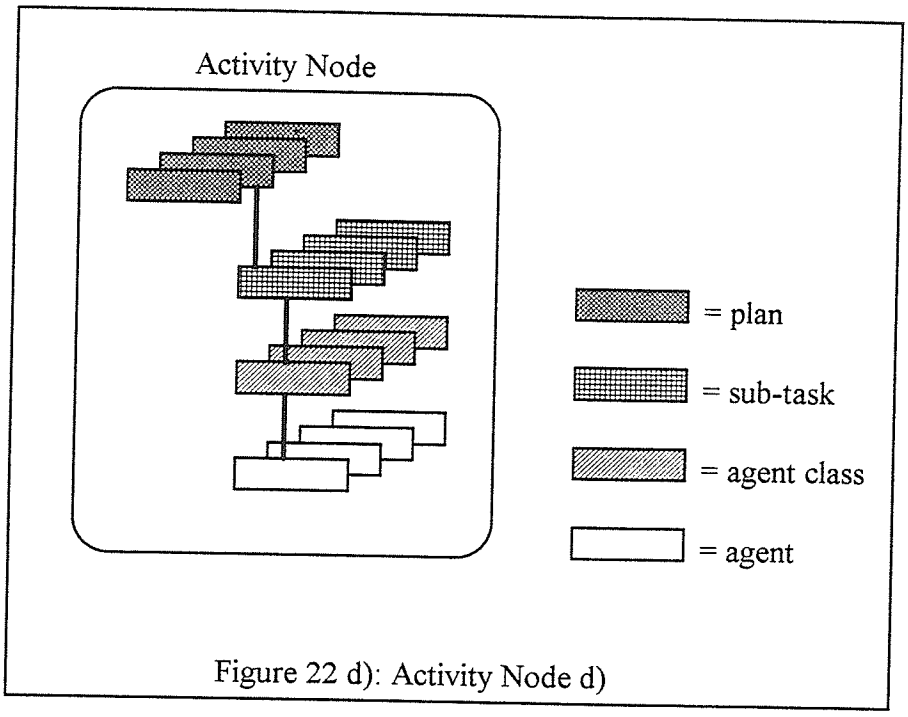


Figure 22 d): Activity Node d)

5.2.4 Process Notifications

Process-notifications can be divided into two main modules: *process-incoming-notification* and *process-outgoing-notification* (see Figure 23).

Process-incoming-notification processes all the new incoming notifications. The corresponding activity node of a notification is found by executing *find-activity-by-req-id*. Unique ids are used to link activity nodes with the requests and notifications. By providing a request id, the corresponding activity node can be found and vice versa. If the incoming notification indicates that the agent has no problem in completing a task, then status of the activity node associated with the notification will be set to *no-problem*. On the other hand, if the incoming notification indicates that the agent cannot complete the task, the corresponding activity node's status will be set to *problem* and a clean-up request will be sent to the original requesting agent.

Process-outgoing-notification generates the required notifications to other agents. It finds all the root activity nodes by executing *find-root-activity-node*. If the status of the root activity node is *no-problem*, then it will send a notification to the original requesting

agent by executing *find-request-given-id* to find the original request and *send-notification* to send out the notification.

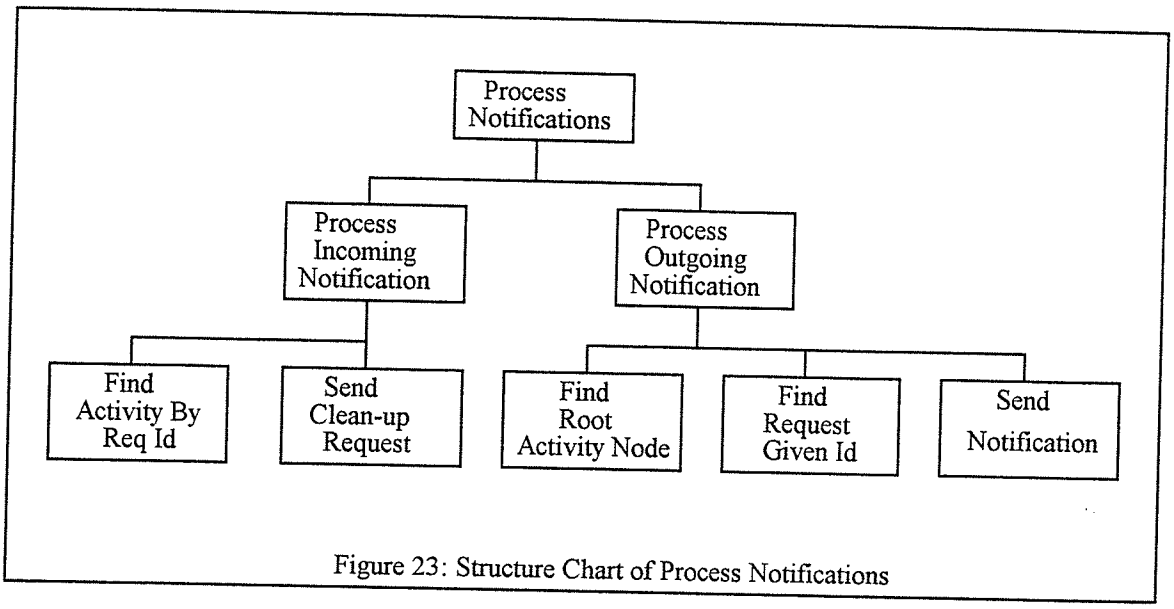


Figure 23: Structure Chart of Process Notifications

5.3 Summary

The system described in this chapter is not a complete implementation of the multiagent planning model discussed in Chapter 4. By using proper knowledge representations and methods, we are able to simulate different ways for agents to reason about their local activities and cooperatively coordinate global activities with other agents. The activities of the system are divided into cycles. By stepping through each cycle, we are able to query the system about the current status of each agent. A cycle constraint mechanism is also implemented in the agents' task constraint list to require a specific number of cycles for an agent to complete a particular task. This allows the user to examine the effects of varying the time it takes some agents to perform tasks.

In the next chapter, we will examine a variety of sample runs of the system using the house building application. Different cases will be discussed to illustrate the flexibility of the prototype system.

Chapter Six : Sample System Runs

In this chapter, we examine several simulations of the house building application to illustrate the operation of the prototype system. Since it is difficult (if not impossible) to illustrate every component of the system, only significant features of the system are presented. The simulations were run in Macintosh Common Lisp. Seven agents were created including the highest level agent USER which is used to initiate the test requests. The other six agents represent different experts required to build major components of a hypothetical house.

6.1 Agent Knowledge

The knowledge base of each agent is setup using several sub-routines. An agent may be defined in more than one sub-routine because under different situations we may want the agent to behave differently. This is accomplished by varying its plan, task, or agent knowledge. Appendix B shows all the sub-routines used in this chapter. Note that a sub-routine like *define-dataset-a* is used to setup a complete set of agents, while a sub-routine like *setup-agent1-a* is used to setup the knowledge of an individual agent (*Mark-House-Building*):

```
(defun setup-agent1-a ()
   (setq agent-1 (MAKE-AGENT :agent-name 'Mark-House-Building
                     :plan-list (LIST
                                 (MAKE-PLAN :plan-name 'Plan-Mark-House-Building
                                            :task-name 'House-Building
                                            :task-list '(Build-Exterior Build-Interior))
                                )
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Build-Exterior
                                            :agent-class-list '(Mark-Class-Build-Exterior))
                                 (MAKE-TASK :task-name 'Build-Interior
                                            :agent-class-list '(Mark-Class-Build-Interior))
                                )
                     :agent-class-list
                                 (LIST
                                  (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Exterior
                                                    :agent-list '(John-House-Building))
                                  (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Interior
                                                    :agent-list '(John-House-Building)))
                     )
    )
)
```

In this example, the name of the agent is *Mark-House-Building*. There is only one plan in the agent's *plan-list* - *Plan-Mark-House-Building*. The associated task name is *House-Building*. This plan specifies that the *House-Building* task can be broken down into two sub-tasks: *Build-Exterior* and *Build-Interior*. Only one class of agents can be considered to work on the *Build-Exterior* task - *Mark-Class-Build-Exterior*. Similarly, only one class of agents is known to be able to work on the *Build-Interior* task - *Mark-Class-Build-Interior*. These two agent classes are defined in the *agent-class-list*. According to the agent knowledge in this agent, only one agent is known to belong to *Mark-Class-Build-Exterior* - *John-House-Building*. Similarly, only one agent is included in the class *Mark-Class-Build-Interior* and it is also *John-House-Building*.

The knowledge associated with other agents are defined in the same way but with different built-in plan, task, and agent knowledge.

6.2 Sample Runs

Sample runs of the system were performed to test different situations that may arise while the agents attempt to solve the House Building problem cooperatively. All the runs are initiated with the same request (*House-Building*) sent from the *User* agent to the *Mark-House-Building* agent. By varying the knowledge in one or more agents, different situations will occur and we can examine how the system responds. The actual print out of each sample run is found in Appendix C.

6.2.1 Dataset A

Dataset A represents a smooth run of the system. Once the plans, tasks, agent classes and agents are chosen, all the agents have no problems performing the required tasks.

After *User* sends the request to *Mark-House-Building*, the system starts processing one agent at a time in a series of cycles.

System Cycle 1:

```
Processing agent: Mark-House-Building ...
Processing request (G464) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION
```

The above statements represent the processing associated with the *Mark-House-Building* agent on the first cycle. *Mark-House-Building* responds to the incoming *House-Building* request by building task nodes in its activity blackboard and setting the status of the task nodes *House-Building, Build-Exterior* and *Build-Interior* to *Awaiting Subtask Distribution, Awaiting Distribution* and *Awaiting Distribution* respectively[11].

On the next cycle, *Mark-House-Building* starts sending the sub-tasks *Build-Exterior* and *Build-Interior* to the *John-House-Building* agent. After *Mark-House-Building* puts the requests in *John-House-Building*'s incoming request queue, *John-House-Building* starts processing the requests by creating task nodes in its activity blackboard and setting the status of the task nodes appropriately. Note that the sub-task *Decoration* can be done by *John-House-Building* itself and it requires 24 simulation cycles. *John-House-Building* sets the Decoration cycle count to 1 at this point. *John-House-Building* also sends sub-task *Lay-Foundation* to *Tom-House-Building* and *Build-House-Frame* to *Paul-House-Building*. The execution statements of *Mark-House-Building* and *John-House-Building* are shown below.

[11] A node with the status *Awaiting Subtask Distribution* corresponds a composite task, while a node with the status *Awaiting Distribution* corresponds to a primitive task.

System Cycle 2:

```
Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G466) of agent (John-House-Building): Task = Build-Exterior
Processing request (G467) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)
```

On the same cycle, *Tom-House-Building* and *Paul-House-Building* also process their incoming requests by creating task nodes in the activity blackboard and setting the proper status[12].

On the next cycle, *John-House-Building* sends another two requests, *Plumbing* and *Electricity*, to *Tom-House-Building*. *Tom-House-Building* processes the requests in the same way as described before. The execution statements are shown below.

System Cycle 3:

```
Processing agent: Tom-House-Building ...
Processing request (G474) of agent (Tom-House-Building): Task = Plumbing
Processing request (G475) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
    Set task: Electricity status to AWAITING SUBTASK DISTRIBUTION
    Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
    Set task: Interior-Electricity status to AWAITING DISTRIBUTION
Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)
```

In addition, *Tom-House-Building* sends requests to perform the *Init-Lay-Foundation* tasks and *Final-Lay-Foundation* to *Doug-House-Building*. It also sends a *Interior-Plumbing* request to *Rick-House-Building*. *Paul-House-Building* sends requests

[12]This occurs on the same cycle because these agents are polled after *John-House-Building*.

to perform the *Build-Roof* and *Build-Wall* tasks to *Rick-House-Building*. *Doug-House-Building* and *Rick-House-Building* process their incoming requests in the same way as other agents. They also start processing the *Init-Lay-Foundation*, *Final-Lay-Foundation*, *Interior-Plumbing*, *Build-Roof* and *Build-Wall* tasks by initializing the cycle count to 1 for these tasks.

On the next cycle, *Tom-House-Building* sends a request to perform the *Interior-Electricity* task to *Rick-House-Building*. *Rick-House-Building* starts processing the incoming request *Interior-Electricity* by setting the node's cycle count to 1 as shown in the execution statements below.

System Cycle 4:

```
Processing agent: Rick-House-Building ...
Processing request (G492) of agent (Rick-House-Building): Task = Interior-Electricity
    Set task: Interior-Plumbing cycle to 2 (Max = 25)
    Set task: Build-Roof cycle to 2 (Max = 25)
    Set task: Build-Wall cycle to 2 (Max = 25)
    Set task: Interior-Electricity cycle to 1 (Max = 25)
```

The next 19 cycles represent the time required by some of the agents to work on the tasks they received. After these 19 cycles, *Doug-House-Building* finishes the *Init-Lay-Foundation* and *Final-Lay-Foundation* tasks without any problems. It then sends a notification to the agent (*Tom-House-Building*) that requested these tasks be performed indicating that the tasks were completed successfully. The execution statements are shown below.

System Cycle 24:

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM
Notification sent from Doug-House-Building to Tom-House-Building: Init-Lay-Foundation
(No-Problem)
Notification sent from Doug-House-Building to Tom-House-Building: Final-Lay-Foundation
(No-Problem)
```

On the next cycle, *John-House-Building* finishes the *Decoration* task successfully. *Tom-House-Building* also sets the status of the *Init-Lay-Foundation* and *Final-Lay-Foundation* task nodes to NO PROBLEM after receiving the notifications from *Doug-House-Building*. The execution statements are shown below.

System Cycle 25:

```
Processing agent: John-House-Building ...
    Set task: Decoration status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM
```

On the next cycle, *Tom-House-Building* sets the *Lay-Foundation* task node's status to NO PROBLEM since both of its sub-tasks, *Init-Lay-Foundation* and *Final-Lay-Foundation,* were completed successfully. A notification is sent to *John-House-Building* which originally requested that the *Lay-Foundation* task be performed to indicate that the *Lay-Foundation* task was completed successfully

System Cycle 26:

```
Processing agent: Tom-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Lay-Foundation
(No-Problem)
```

On the next cycle, *Rick-House-Building* finishes the *Interior-Plumbing*, *Build-Roof* and *Build-Wall* tasks without incidence. It sends notifications to *Tom-House-Building* and *Paul-House-Building* as shown below.

System Cycle 27:

```
Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM
    Set task: Interior-Electricity cycle to 25 (Max = 25)
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Plumbing (No-
Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Roof (No-Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Wall (No-Problem)
```

Tom-House-Building then sets the status of its *Interior-Plumbing* status to NO PROBLEM after receives the notification from *Rick-House-Building*. Similarly, *Paul-House-Building* sets the status of its *Build-Roof* and *Build-Wall* task nodes to NO PROBLEM after receiving the notifications from *Rick-House-Building*. *Rick-House-Building* also completes the *Interior-Electricity* task during this cycle. It sends a notification back to *Tom-House-Building* indicating that the task was completed successfully. The execution statements are shown below.

System Cycle 28:

```
Processing agent: Tom-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity status to NO PROBLEM
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Electricity
(No-Problem)
```

Tom-House-Building then sets the status of the *Plumbing* and *Interior-Electricity* task nodes to NO PROBLEM. It also sends a notification back to *John-House-Building* indicating that this task has been completed. During the same cycle, *Paul-House-Building* sets the status of the *Build-House-Frame* task node to NO PROBLEM. It also sends a

notification back to *John-House-Building* indicating that task has been completed. The execution statements are shown below.

System Cycle 29:

```
Processing agent: Tom-House-Building ...
    Set task: Plumbing status to NO PROBLEM
    Set task: Interior-Electricity status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Plumbing (No-Problem)

Processing agent: Paul-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Build-House-Frame
(No-Problem)
```

John-House-Building then sets the status of its *Plumbing* and *Build-House-Frame* task nodes to NO PROBLEM. *Tom-House-Building* sets the status of its *Electricity* task node to NO PROBLEM. It also sends a notification to *John-House-Building* indicating that this task has been finished. The execution statements are shown below.

System Cycle 30:

```
Processing agent: John-House-Building ...
    Set task: Plumbing status to NO PROBLEM
    Set task: Build-House-Frame status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Electricity status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Electricity
(No-Problem)
```

As a result, *John-House-Building* sets the status of its *Build-Exterior* and *Electricity* task nodes to NO PROBLEM. It also sends a notification to *Mark-House-Building* indicating that the *Build-Exterior* task has been completed. The execution statements are shown below.

System Cycle 31:

```
Processing agent: John-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM
    Set task: Electricity status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Exterior
(No-Problem)
```

Mark-House-Building then sets the status of its *Build-Exterior* task node to NO PROBLEM. *John-House-Building* sets the *Build-Interior* task node's status to NO PROBLEM. It also sends a notification to *Mark-House-Building* indicating that the task has been completed. The execution statements are shown below.

System Cycle 32:

```
Processing agent: Mark-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Interior
(No-Problem)
```

On the next cycle, *Mark-House-Building* sets the *Build-Interior* task node's status to NO PROBLEM as shown below.

System Cycle 33:

```
Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
```

Finally, *Mark-House-Building* sets the status of the *House-Building* task to NO PROBLEM. It also sends a notification to the originator of the *House-Building* task request (*User*) indicating that the task has been completed successfully. At this point, the *House-Building* problem has been solved. The execution statements are shown below.

System Cycle 34:

```
Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO PROBLEM
Notification sent from Mark-House-Building to User: House-Building (No-Problem)
```

In order to analyze the system's functionality, the user can examine the internal data structures of each agent at the end of any simulator cycle as shown below.

```
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
   Incoming Notification:
        John-House-Building:  Build-Exterior  (No-Problem)
        John-House-Building:  Build-Interior  (No-Problem)
   Outgoing Notification:
        User:  House-Building  (No-Problem)
   Activity Blackboard:
     Task: House-Building  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Exterior  Status: No-Problem  Current Agent: John-House-Building
     Task: Build-Interior  Status: No-Problem  Current Agent: John-House-Building
```

In this example, we have displayed information associated with the *Mark-House-Building* agent. It shows the planning knowledge, task knowledge and agent knowledge as described in the previous chapter. The incoming requests list, outgoing requests list, incoming notifications list, outgoing notifications list and the activity blackboard show the activities the agent has performed.

6.2.2 Dataset B1

Dataset B1 represents a simulation in which one agent fails to perform a task, forcing the originator of the task request to send the same task to another agent. The second agent performs the task successfully. The main purpose of this example is to demonstrate the ability of the system to enable an agent to send a task to an alternative agent when one agent fails to solve it.

The first 2 cycles of the system are identical to those shown in section 6.2.1. In cycle 3, *Tom-House-Building* indicates that it has problem performing the *Electricity* task by setting the task status to NO APPLICABLE PLANS. It then sends a notification back to the agent that initiated the task request, *John-House-Building*, indicating that the task cannot be solved. The execution statements are shown below.

System Cycle 3:

```
Processing agent: Tom-House-Building ...
Processing request (G163) of agent (Tom-House-Building): Task = Plumbing
Processing request (G164) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
Notification sent from Tom-House-Building to John-House-Building: Electricity
(Task-Problem)
```

On the next cycle, *John-House-Building* sends a request to *Tom-House-Building* asking it to clean-up the *Electricity* task. *Tom-House-Building* responds by deleting the node on its activity blackboard. The execution statements are shown below.

System Cycle 4:

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 3 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...
Processing request (G164) of agent (Tom-House-Building): Clean-Up = Electricity
Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)
```

John-House-Building then examines alternative agents in its agent knowledge and sends the *Electricity* task to *Paul-House-Building*. *Paul-House-Building* receives the task and starts working on it as shown below.

System Cycle 5:

```
Processing agent: John-House-Building ...
Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
    Set task: Decoration cycle to 4 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G164) of agent (Paul-House-Building): Task = Electricity
    Set task: Electricity cycle to 1 (Max = 23)
```

The rest of the simulation is completed without incidence as described in section 6.2.1.

6.2.3 Dataset B2

Dataset B2 represents a simulation in which one agent fails to perform a task and the task is sent to another agent in the same agent class. This example is similar to the previous one; however, this time the second agent also has problem in performing the task. There are no other alternative agents in the same agent class and no alternative agent classes that the agent can use. Consequently, the entire *House-Building* problem cannot be solved.

The first 4 cycles are the same as those in section 6.2.2. During the fifth cycle, *John-House-Building* sends the *Electricity* task, that could not be completed by *Tom-House-Building,* to *Paul-House-Building*. This time *Paul-House-Building* also has problems in performing the task as shown below.

System Cycle 5:

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G216) of agent (Paul-House-Building): Task = Electricity
    Set task: Electricity status to NO APPLICABLE PLANS
```

On the next cycle, *Paul-House-Building* sends a notification back to *John-House-Building* indicating that it cannot perform the *Electricity* task as shown below.

System Cycle 6:

```
Processing agent: Paul-House-Building ...
Notification sent from Paul-House-Building to John-House-Building: Electricity
(Task-Problem)
```

John-House-Building responds by sending a clean-up request to *Paul-House-Building* for the *Electricity* task. *Paul-House-Building* performs the clean-up job as shown below.

System Cycle 7:

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G216) of agent (Paul-House-Building): Clean-Up = Electricity
```

On the next cycle, *John-House-Building* determines that it has no alternate method of solving the *Electricity* task. As a result, it sets the status of the *Electricity* task to NO APPLICABLE PLANS as shown below.

System Cycle 8:

```
Processing agent: John-House-Building ...
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 7 (Max = 24)
```

John-House-Building then realizes that it has a problem in solving the *Build-Interior* task due to the failure of *Electricity* task.

System Cycle 9:

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
    Set task: Decoration cycle to 8 (Max = 24)
```

John-House-Building then determines that it has no alternate method for solving the *Build-Interior* task as shown below.

System Cycle 10:

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 9 (Max = 24)
```

On the next cycle, *John-House-Building* sends a notification back to the requesting agent (*Mark-House-Building*) indicating that it has a problem in performing the *Build-Interior* task as shown below.

System Cycle 11:

```
Processing agent: John-House-Building ...
Notification sent from John-House-Building to Mark-House-Building: Build-Interior
(Task-Problem)
    Set task: Decoration cycle to 10 (Max = 24)
```

Mark-House-Building responds by sending a clean-up request to *John-House-Building* to abort the *Build-Interior* task. This leads to a chain reaction of clean-up jobs. *John-House-Building* sends a clean-up request to *Tom-House-Building* to clean-up the *Plumbing* task even though *Tom-House-Building* does not report any problem in performing the *Plumbing* task. This is required because *Plumbing* is a sub-task of *Build-Interior*. When the *Build-Interior* task is aborted, all of its children tasks must be canceled. The execution statements are shown below.

System Cycle 12:

```
Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G208) of agent (John-House-Building): Clean-Up = Build-Interior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)
```

Subsequent clean-up requests are processed until the entire *House-Building* task has been aborted and the *User* agent has been notified.

6.2.4 Dataset C1

Dataset C1 represents a simulation in which one agent fails to perform a task causing the requesting agent to send the same task to another agent in a different agent class. A different agent class is chosen because there is no alternative agent available in the original agent class and an alternative agent class for the same task is available. The agent from the new agent class has no problem in performing the task. The main purpose of this

example is to demonstrate the ability of the system to choose an alternative agent class for a task when the agents from the initial agent class selected to perform the task encounter difficulties.

The first 2 cycles are identical to those shown in section 6.2.1. In cycle 3, *Tom-House-Building* indicates that it has a problem in performing the *Electricity* task. As a result, it sends a notification back to *John-House-Building* as shown below.

System Cycle 3:

```
Processing agent: Tom-House-Building ...
Processing request (G270) of agent (Tom-House-Building): Task = Plumbing
Processing request (G271) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
Notification sent from Tom-House-Building to John-House-Building: Electricity
(Task-Problem)
```

On the next cycle, *John-House-Building* sends a clean-up request to *Tom-House-Building* which processes the clean-up request as shown below.

System Cycle 4:

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 3 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...
Processing request (G271) of agent (Tom-House-Building): Clean-Up = Electricity
Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)
```

John-House-Building then sends the *Electricity* task that *Tom-House-Building* failed to solve to *Paul-House-Building*, using an alternative agent class from its agent class knowledge associated with the *Electricity* task. The execution statements are shown below.

System Cycle 5:

```
Processing agent: John-House-Building ...
Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
    Set task: Decoration cycle to 4 (Max = 24)
```

The remainder of the simulation is completed successfully as in section 6.2.1.

6.2.5 Dataset D1

Dataset D1 represents a simulation in which an agent fails to perform a task and the requesting agent is forced to abort the initial plan and apply an alternative plan. A different plan is chosen because there is no alternative agent available in the original class and there are no alternative agent classes available in the original plan. The new plan decomposes the original problem into a different set of tasks which are performed successfully.

The first 2 cycles are similar to those in section 6.2.1. In cycle 3, *Tom-House-Building* indicates that it has problem in performing *Plumbing* as shown below.

System Cycle 3:

```
Processing agent: Tom-House-Building ...
Processing request (G377) of agent (Tom-House-Building): Task = Plumbing
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to NO APPLICABLE PLANS
```

On the next cycle, *Tom-House-Building* sends a notification back to *John-House-Building* indicating that it cannot perform the *Plumbing* task as shown below.

System Cycle 4:

```
Processing agent: Tom-House-Building ...
Notification sent from Tom-House-Building to John-House-Building: Plumbing (Task-Problem)
```

John-House-Building responds by sending a clean-up request to *Tom-House-Building* to abort its *Plumbing* task. *Tom-House-Building* processes the clean-up of the *Plumbing* task as shown below.

System Cycle 5:

```
Processing agent: John-House-Building ...
    Set task: Plumbing status to TASK PROBLEM
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G377) of agent (Tom-House-Building): Clean-Up = Plumbing
```

John-House-Building then indicates that it has no alternative plans for the *Plumbing* task as shown below.

System Cycle 6:

```
Processing agent: John-House-Building ...
    Set task: Plumbing status to NO APPLICABLE PLANS
```

On the next cycle, *John-House-Building* indicates that it has problem in performing *Build-Interior* task as a result of the inability to complete its subtask - *Plumbing*.

System Cycle 7:

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
```

On the next cycle, *John-House-Building* temporarily sets the status of *Build-Interior* task node to NO APPLICABLE PLANS indicating that the current plan has been aborted.

System Cycle 8:

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
```

At this point, *John-House-Building* checks to see if the current plan is the last available plan in the plan knowledge. If it is not the last plan, then *John-House-Building* selects an alternative plan for the *Build-Interior* task. Instead of using the *Plan-John-Build-Interior-A* plan which includes *Plumbing*, it uses the *Plan-John-Build-Interior-B* plan which includes *Electricity* and *Decoration*. *John-House-Building* can do the *Decoration* task by itself. But the *Electricity* task must be sent to another agent as shown below.

System Cycle 9:

```
Processing agent: John-House-Building ...
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
```

On the next cycle, *John-House-Building* sends the *Electricity* task to *Tom-House-Building* and *Tom-House-Building* processes the incoming task as shown below.

System Cycle 10:

```
Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G398) of agent (Tom-House-Building): Task = Electricity
    Set task: Electricity status to AWAITING SUBTASK DISTRIBUTION
    Set task: Interior-Electricity status to AWAITING DISTRIBUTION
```

The remainder of simulation is completed without incidence similar to the processing in section 6.2.1.

6.2.6 Dataset D2

The final example, Dataset D2, represents a simulation in which an agent fails to perform a task and the requesting agent is forced to abort the initial plan and apply an alternative plan. A different plan is chosen because there is no alternative agent available in the original agent class and there is no alternative agent class available in the original plan. This situation is the same as in the previous section, but this time the alternative plan also fails to perform the task. In this case, the entire *House-Building* problem fails because there are no other alternative plans for the *House-Building* task.

The first nine cycles are the same as those in section 6.2.6. On the tenth cycle, *John-House-Building* sends the *Electricity* task to *Paul-House-Building* based on an alternative plan. *Paul-House-Building* indicates that it has problem in performing the *Electricity* task as shown below.

System Cycle 10:

```
Processing agent: John-House-Building ...
Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G447) of agent (Paul-House-Building): Task = Electricity
    Set task: Electricity status to NO APPLICABLE PLANS
```

On the next cycle, *Paul-House-Building* sends a notification back to *John-House-Building* indicating that it cannot perform the *Electricity* task as shown below.

System Cycle 11:

```
Processing agent: Paul-House-Building ...
Notification sent from Paul-House-Building to John-House-Building: Electricity
(Task-Problem)
```

John-House-Building responds by sending a clean-up request to *Paul-House-Building* to abort the *Electricity* task. *Paul-House-Building* processes the clean-up request as shown below.

System Cycle 12:

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 4 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G447) of agent (Paul-House-Building): Clean-Up = Electricity
```

On the next cycle, *John-House-Building* indicates that there are no alternative plans for the *Electricity* task as shown below.

System Cycle 13:

```
Processing agent: John-House-Building ...
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 5 (Max = 24)
```

As a result, a series of clean-up requests are distributed which eventually results in the initial *House-Building* task being aborted.

6.3 Summary

The example simulations in this chapter do not cover all the possible situations that the system may face. They do, however, illustrate situations where an agent needs to send a task to another agent from the same agent class, from a different agent class, or using a different plan. The user can examine the internal data structures, including the entire agent structure and the agent's activity blackboard, of each agent at the end of any simulator cycle. The data structure information can be displayed by choosing options from the menus in the system.

The system can be easily modified to include additional agents and more complete plan, task and agent knowledge for each agent. Additional applications can also be simulated using different agent knowledge.

Chapter Seven : Conclusions

In the classical single agent planning model, problems are posed to a planner as an initial state and a goal state description. The initial state describes the way the world is at the initial point in time. The goal state describes the way we want the world to look when the plan has been executed. The world refers to the application domain where the planning takes place. The planner tries to find a plan (or a set of actions) such that by executing the plan from the initial state of the problem, the goal state will be generated. Planning is essentially a search problem. The planner must traverse a potentially large search space and find a plan that is applicable in the initial state and produces the goal state when executed. The complexity of the planning system grows when several potentially applicable plans can be used. In many real world problems this results in a combinatorial explosion of alternatives that overwhelms most single agent planning systems.

The classical definition of the planning problem typically employs a state-based representation of the world. The world is represented by taking a "snapshot" of it at one time and describing the world as it appears in this snapshot. The planner assumes that the initial state of the world does not change while the plan is executing. Thus, the planner constructs a plan based on the initial state of the world without reacting to any changes that might occur during the planning process. This is an important limitation in classical planners. This also leads to the distinction between plan time and execution time. Classical planners also cannot react to the changes that occur while the system is executing a plan.

Multiagent planning research studies how a loosely coupled network of problem solvers (or agents) can work together to solve problems that are beyond their individual capabilities. Each agent in the network is capable of solving sophisticated problems using its own expertise and can work independently. Many of the problems faced by the group of agents cannot be completed without cooperation among the agents. Cooperation is

necessary because no single agent has enough expertise, resources, and information to solve a particular problem. Agents must rely on each other to solve the problem.

We have described an architecture that supports the basic multiagent planning actions of task decomposition, task distribution, and result integration through the use of activity and agent modeling. Control among agents may be centralized, partially centralized, or completely distributed depending on the characteristics of the application. The architecture also provides adaptive planning when incompatibilities arise.

A simulation system based on our model of multiagent planning was built using Macintosh Common Lisp. The simulator provides a test bed for exploring various multiagent planning techniques and issues. The test bed allows us to simulate various ways that agents can reason about their local activities and cooperatively coordinate global activities with other agents.

The constraint directed components of the multiagent planning model have not been completely implemented. Data structure extensions and algorithms for processing constraints should be added in the future to provide general constraint functions. Constraints can be used to assist agents in selecting plans, agent classes, and agents. Constraints can be added to the plan knowledge. This can assist the agent in choosing a suitable plan. The program model that is responsible in selecting a plan can use these constraints. The ways that constraints are distributed among agents must be considered. The initial implementation was built with the addition of constraint processing in mind. Qualitative constraints (e.g., the material quality must be grade A) can be distributed to the agents without modification. For quantitative constraints (e.g., the time required to finish the entire project must be less than 100 days) a method of dividing the constraints among the agents must be introduced. This decomposition information could be added to the plan knowledge of an agent.

The system presented in Chapter 6 addresses a specific application - *House Building*. By modifying the knowledge base of the agents, the system can be used in other

applications. Such separation of data and code is a fundamental principle of AI research and provides ease of modification of the system. A more complete object-oriented approach may also be used to implement the system such that data and methods are bound together in an agent[14]. An agent can be represented as an object. Different agent classes can be represented by different object classes. New agents could then be easily created with basic properties inherited from an agent class.

As real world problems get more and more complicated, more sophisticated planning systems are required. Single agent planning systems are not capable of solving many real world problems. A multiagent planning system provides a more dynamic way to solve complicated problems that involve the coordinated integration of different agents' expertise.

[14]The *defstruct* construct used in the prototype does not provide the sophisticated object-oriented facilities found in the Common Lisp Object System (CLOS).

Appendix A : References

[Adler and Simoudis, 1990] Adler, Mark R. and Simoudis, Evangelos, Integrating Distributed Expertise, 10th AAAI International Workshop on Distributed Artificial Intelligence, Oct 1990.

[Barr, Cohen and Feigenbaum, 1989] Barr, A., Paul R. Cohen, and Edward A. Feigenbaum, The Handbook of Artificial Intelligence, Volume IV, in Addison-Wesley Publishing Company, Inc., 1989.

[Bond and Gasser, 1988] Bond, Alan H., and Les Gasser, Readings in Distributed Artificial Intelligence, in Morgan Kaufmann Publishers, Inc., 1988.

[Cohen and Feigenbaum, 1989] Cohen, Paul R., and Feigenbaum, Edward A., The Handbook of Artificial Intelligence, Volume III, in Addison-Wesley Publishing Company, Inc., 1989.

[Conry, Meyer and Lesser, 1986] Conry, Susan E., Meyer, Robert A., and Lesser, Victor R., Multistage Negotiation in Distributed Planning, COINS Technical Report 1986-1987, Dec 1986.

[Corkill, 1979] Corkill, Daniel D., Hierarchical Planning in a Distributed Environment, in the 6th IJCAI, 1979.

[Davis and Smith, 1988] Davis, Randall, and Smith, Reid G., Negotiation as a Metaphor for Distributed Problem Solving, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc, 1988.

[Durfee, Lesser and Corkill, 1989] Durfee, Edmund H., Victor R. Lesser, and Daniel D. Corkill, Trends in Cooperative Distributed Problem Solving, in IEEE Transactions on Knowledge and Data Enginerring, Vol. 1, No. 1, March 1989.

[Durfee and Montgomery, 1990] Durfee, Edmund H., and Montgomery, Thomas A., A Hierarchical Protocol for Coordinating Multiagent Behaviors, 10th AAAI International Workshop on Distributed Artificial Intelligence, Oct 1990.

[Engelmore and Morgan, 1988] Engelmore, R., and Tony Morgan, Blackboard Systems, in Addison-Wesley Publishing Company, 1988.

[Ernst and Newell, 1969] Ernst, G. and Newell, A., A Case Study in Generality and Problem Solving, ACM Monograph Series, Academic Press, New York, 1969.

[Evans, 1988] Evans, M., A Knowledge-Based Model of Distributed Problem Solving, Ph.D. Dissertation, Computer Science, University of Manitoba.

[Evans and Anderson, 1989] Evans, M., and J. Anderson, A Constraint-Directed Architecture for Multi-Agent Planning, in Proceedings of the Ninth AAAI Distributed Artificial Intelligence Workshop, Seattle, WA, 1989.

[Evans and Anderson, 1990] Evans, M., and J. Anderson, An Analysis of Constraints for Multi-Agent Problem Solving, Technical Report, Department of Computer Science, University of Manitoba, February, 1990.

[Evans, Anderson and Crysdale, 1992] Evans, M., J. Anderson, and G. Crysdale, Achieving Flexible Autonomy in Multi-Agent Systems using Constraints, in Applied Artificial Intellegence, Spring 1992.

[Fikes and Nilsson, 1971] Fikes, Richard E., and Nils J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, in Artificial Intelligence 2, 1971.

[Fox, 1988] Fox, Mark S., An Organizational View of Distributed Systems, Readings in Distributed Artificial Intelligence, in Morgan Kaufmann Publishers, Inc., 1988.

[Hayes-Roth, 1988] Hayes-Roth, Barbara, A Blackboard Architecture for Control, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc, 1988.

[Hendler, Tate and Drummond, 1990] Hendler, J., Austin Tate, and Mark Drummond, AI Planning: Systems and Techniques, in AAAI, 1990.

[Huhns, Bridgeland and Arni, 1990] Huhns, Michael N., Bridgeland, David Murray, and Arni, Natraj Vidur, A DAI Communication Aide, 10th AAAI International Workshop on Distributed Artificial Intelligence, Oct 1990.

[Morgenstern, 1988] Morgenstern, Leora, Knowledge Preconditions for Actions and Plans, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc, 1988.

[Noyes, 1992] Noyes, James L., Artificial Intelligence with Common Lisp, D. C. Heath and Company, 1992.

[Sacerdoti, 1974] Sacerdoti, Earl D., Planning in a Hierarchy of Abstraction Spaces, in Artificial Intelligence 5, 1974.

[Sacerdoti, 1975] Sacerdoti, Earl D., The Nonlinear Nature of Plans, in the 4th IJCAI, 1975.

[Shaw, 1990] Shaw, Michael J., Mechanisms for Cooperative Problem Solving and Multi-Agent Learning in Distributed Artificial Intelligence Systems, 10th AAAI International Workshop on Distributed Artificial Intelligence, Oct 1990.

[Smith, 1988] Smith, Reid G., The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc, 1988.

[Smith and Davis, 1988] Smith, Reid G., and Davis, Randall, Frameworks for Cooperation in Distributed Problem Solving, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc, 1988.

[Tate, 1977] Tate, A., Generating Project Networks, in the 5th IJCAI, 1977.

[Wilkins, 1988] Wilkins, David E., Practical Planning: Extending the Classical AI Planning Paradigm, in Morgan Kaufmann Publishers, Inc., 1988.

Appendix B : Data Structures and Program Routines

Data Structures:

```
;
; Global list of agent-names and agent-ptrs
;
(defvar *SYSTEM-AGENT-LIST*)

;
; plan structure
;
(defstruct PLAN
  plan-name
  task-name
  task-constraints
  task-list)

;
; task structure
;
(defstruct TASK
  task-name
  task-constraints
  agent-class-list)

;
; agent class structure
;
(defstruct AGENT-CLASS
  class-name
  agent-list)

;
; agent structure
;
(defstruct AGENT
  agent-name
  plan-list
  task-list
  agent-class-list
  constraint-list
  incoming-request-list
  outgoing-request-list
  incoming-notification-list
  outgoing-notification-list
  activity-blackboard
  (processing-constraints (MAKE-PROCESSING-CONSTRAINTS)))
```

```
;
; request structure
;
(defstruct REQUEST
  from-agent
  to-agent
  (request-id (gensym))
  (request-status 'initiated)
  (request-type 'task)
  task-name
  activity-node-id
  request-args)


;
; notification structure
;
(defstruct NOTIFICATION
  from-agent
  to-agent
  (notif-id (gensym))
  (notif-status 'just-received)
  (notif-type 'result)
  request-id      ; optional
  notif-info)


;
; activity structure
;
(defstruct ACTIVITY
  task
  task-args
  (Activity-id (gensym))
  (status 'holding)
  (cycle 0)
  (activity-type 'plan)
  applicable-agent-classes
  current-agent-class
  applicable-agents
  current-agent
  parent-task-id
  parent-request-id
  applicable-plans-and-tasks
  current-plan-or-task
  current-task-list)
```

Program Routines:

"The following routine returns the agent structure associated with a given agent name. It returns NIL if the given agent name does not exist. The global agent list stored in *SYSTEM-AGENT-LIST* is searched."

```
(defun FIND-AGENT (agent-name)
  (eval (second (assoc agent-name *SYSTEM-AGENT-LIST*)))
)
```

"The following routine accepts an agent structure and a request structure. It adds the request structure to the list of incoming requests for the agent."

```
(defun ADD-INCOMING-REQUEST (agent request)
  (setf (AGENT-incoming-request-list agent)
        (append (AGENT-incoming-request-list agent) (list request)))
)
```

"The following routine accepts an agent structure and a request structure. It adds the request structure to the list of outgoing requests for the agent."

```
(defun ADD-OUTGOING-REQUEST (agent request)
  (setf (AGENT-outgoing-request-list agent)
        (append (AGENT-outgoing-request-list agent) (list request)))
)
```

"The following routine accepts an agent structure and a notification structure. It adds the notification structure to the list of incoming notifications for the agent."

```
(defun ADD-INCOMING-NOTIFICATION (agent notif)
  (setf (AGENT-incoming-notification-list agent)
        (append (AGENT-incoming-notification-list agent) (list notif)))
)
```

"The following routine accepts an agent structure and a notification structure. It adds the notification structure to the list of outgoing notifications for the agent."

```
(defun ADD-OUTGOING-NOTIFICATION (agent notif)
  (setf (AGENT-outgoing-notification-list agent)
        (append (AGENT-outgoing-notification-list agent) (list notif)))
)
```

"The following routine performs the sending of a request from one agent to another. It accepts two agent structures (from and to) and keyword arguments. The incoming and outgoing request lists of the corresponding agents are updated to reflect the communication."

```
(defun SEND-REQUEST (from-agent-name to-agent-name
                     request-type task-name request-args activity-node-id)
  (let* ((to-agent    (FIND-AGENT to-agent-name))
         (from-agent  (FIND-AGENT from-agent-name))
         (new-request (MAKE-REQUEST)))

    (setf (REQUEST-from-agent   new-request) from-agent-name)
    (setf (REQUEST-to-agent     new-request) to-agent-name)
    (setf (REQUEST-request-type new-request) request-type)
    (setf (REQUEST-task-name    new-request) task-name)
    (setf (REQUEST-request-args new-request) request-args)

    (if (not (equal activity-node-id nil))
        (setf (REQUEST-request-id new-request) activity-node-id)
    )

    (PRINT-REQUEST new-request)

    (if (or (null to-agent)
            (null from-agent))
        (progn   ; if to or from agent is not found then error has occured
          (princ "*** Error: destination or source agent not found")
          (terpri)
          (terpri))
        (progn   ; else update request lists in the agents and send notification of
receipt
          (ADD-OUTGOING-REQUEST from-agent new-request)
          (ADD-INCOMING-REQUEST to-agent   new-request)
        )
    )
  )
)
```

"The following routine sends a notification from one agent to another. It accepts a request-id and a notification-type to be assigned to the notification. The incoming and outgoing notification lists of the corresponding agents are updated to reflect the communication."

```
(defun SEND-NOTIFICATION (from-agent-name to-agent-name
                          request-id notif-type notif-info)
  (let ((new-notif  (MAKE-NOTIFICATION))
        (to-agent   (FIND-AGENT to-agent-name))
        (from-agent (FIND-AGENT from-agent-name)))

    (setf (NOTIFICATION-from-agent new-notif) from-agent-name)
    (setf (NOTIFICATION-to-agent   new-notif) to-agent-name)
    (setf (NOTIFICATION-notif-info new-notif) notif-info)
    (setf (NOTIFICATION-notif-type new-notif) notif-type)
    (setf (NOTIFICATION-request-id new-notif) request-id)

    (PRINT-NOTIFICATION new-notif)

    (if (or (null to-agent)
            (null from-agent))
        (progn    ; if to or from agent is not found then error has occured
          (princ "*** Error: destination or source agent not found")
          (terpri)
          (terpri))
        (progn    ; else update notification lists in the agents
          (ADD-OUTGOING-NOTIFICATION from-agent new-notif)
          (ADD-INCOMING-NOTIFICATION to-agent   new-notif))
```

```
    )
  )
)
```

"The following routine prints the contents of a given request sent within the system. The information printed includes: the sender, recepient and request-info."

```
(defun PRINT-REQUEST (r)
  (format t "Request sent from ~A to ~A: ~A (~A)"
    (REQUEST-from-agent    r)
    (REQUEST-to-agent      r)
    (REQUEST-request-type  r)
    (REQUEST-task-name     r))
  (terpri))
```

"The following routine prints the contents of a given notification sent within the system. The information printed includes: the sender, recepient, and notification-info."

```
(defun PRINT-NOTIFICATION (n)
  (format t "Notification sent from ~A to ~A: ~A (~A)"
    (NOTIFICATION-from-agent   n)
    (NOTIFICATION-to-agent     n)
    (NOTIFICATION-notif-type   n)
    (NOTIFICATION-notif-info   n))
  (terpri))
```

"The following routine coordinates the processing cycles for the agents in the system. The routine optionally accepts a processing-constraints structure that can be used to override the constraints associated with the individual agents for the current cycle."

```
(defun PROCESS-AGENTS (&key (constraints nil))
  (let ((agent nil))
    (dolist (a *SYSTEM-AGENT-LIST*)
      (setf agent (FIND-AGENT (first a)))
      (terpri)
      (format t "Processing agent: ~A ..."
            (first a))
      (terpri)
      (PROCESS-AN-AGENT agent
                  :processing-constraints constraints)
    )
  )
)
```

"The following routine coordinates the processing for a given agent for the current processing cycle. The routine optionally accepts a processing-constraint structure that, if present, overrides the agent's internal processing constraints. The routine first invokes PROCESS-REQUESTS to process 0 or more requests received in the incoming request queue on the last or previous cycles and then it invokes PROCESS-ACTIVITY-NODES to process 0 or more activity nodes in the activity blackboard, and finally it invokes PROCESS-NOTIFICATIONS to process 0 or more incoming notifications received in the incoming notification queue on the last or previous cycles.

The processing constraints govern what type and how many requests and notifications can be processed."

```
(defun PROCESS-AN-AGENT (agent &key (processing-constraints nil))
    (PROCESS-REQUESTS        agent
                             processing-constraints)
    (PROCESS-ACTIVITY-NODES agent
                             processing-constraints)
    (PROCESS-NOTIFICATIONS   agent
                             processing-constraints)
)
```

"The following routine processes the newly received requests in the incoming request queue. The special-processing-constraints parameter can be used to override the processing constraints associated with the given agent (maximum number of requests that can be processed on this cycle and the type(s) of tasks that can be processed)."

```
(defun PROCESS-REQUESTS (agent special-processing-constraints)
  (let* ((process-constraints
              (OR special-processing-constraints
                  (AGENT-processing-constraints agent)))
         (request-constraints
                   (PROCESSING-CONSTRAINTS-request-constraints process-constraints))
         (target-requests (FIND-TARGET-REQUESTS agent request-constraints))
         )
    (dolist (r target-requests)
      (PROCESS-INDIVIDUAL-REQUEST agent r)
      )
    )
)
```

"The following routine attempts to build a list of request ids from the given agent's incoming request list and the given request constraints. It ensures that the maximum number of requests set for this cycle is enforced (0 or more) and that each request selected meets the prescribed criteria (type of task for now)."

```
(defun FIND-TARGET-REQUESTS (agent request-constraints)
  (let* ((max-requests
                   (REQUEST-CONSTRAINTS-max-requests request-constraints))
         (request-criteria
                   (REQUEST-CONSTRAINTS-criteria request-constraints))
         (target-requests nil)
         (request-cnt 0))

    (dolist (r (AGENT-incoming-request-list agent))
      (if (= request-cnt max-requests)
        (return))

      (if (ACCEPTABLE-REQUEST r request-criteria)
        (progn
          (setf target-requests (append target-requests
                                        (list r)))
          (setf request-cnt (1+ request-cnt))
        )
       )
    )
    target-requests)
)
```

"The following routine determines whether a given request meets the required criteria for processing. For now the request-criteria is either an empty list indicating any task can be processed or a list of specific tasks that are allowed. In either case the request must be in INITIATED status and must be a TASK or CLEAN-UP request."

```
(defun ACCEPTABLE-REQUEST (request request-criteria)
  (if (and
       (or (equal (REQUEST-request-type request) 'task)
           (equal (REQUEST-request-type request) 'clean-up))
       (equal (REQUEST-request-status request) 'Initiated)
       (or (null request-criteria)
           (member (REQUEST-task-name request) request-criteria)))
    t)
)
```

"The following routine processes a given request for a given agent based on a given request id. If the request is clean-up request, it invokes the clean-up routine. If the request is a task request, it creates a new activity node for it by invoking CREATE-ACTIVITY-NODE-FOR-REQUEST routine."

```
(defun PROCESS-INDIVIDUAL-REQUEST (agent request)
    (format t "Processing request (~A) of agent (~A): ~A = ~A"
        (REQUEST-request-id   request)
        (AGENT-agent-name     agent)
        (REQUEST-request-type request)
        (REQUEST-task-name    request))
    (terpri)

    (if (equal (REQUEST-request-type request) 'CLEAN-UP)
        (CLEAN-UP-ACTIVITY-NODE-FOR-REQUEST agent request)
        (if (not (equal (REQUEST-request-type request) 'CLEANED))
            (CREATE-ACTIVITY-NODE-FOR-REQUEST   agent request)
        )
    )
)
```

"If the request status is INITIATED, the following routine invokes DELETE-ACTIVITY-SUB-TASK-NODES to delete the activity node with the request id equal to the one specified in the request parameter. It also deletes all of its children nodes."

```
(defun CLEAN-UP-ACTIVITY-NODE-FOR-REQUEST (agent request)
    (if (equal (REQUEST-request-status request) 'INITIATED)
        (progn
            (dolist (node (AGENT-activity-blackboard agent))
                (if (and (equal (ACTIVITY-parent-request-id node) (REQUEST-request-id
request))
                         (equal (ACTIVITY-parent-task-id    node) nil))
                    (DELETE-ACTIVITY-SUB-TASK-NODES agent node)
                )
            )
            (setf (REQUEST-request-status request) 'CLEANED)
        )
    )
)
```

"The following routine searches a given agent's incoming request list for the request structure with the given request id. If found it returns the entire request structure else it returns NIL."

```
(defun FIND-REQUEST-GIVEN-ID (agent request-id)
  (dolist (r (AGENT-incoming-request-list agent))
    (if (equal (REQUEST-request-id r) request-id)
      (return r))
    )
)
```

"The following routine creates an activity node for the given request on the given agent's activity blackboard. The new activity node will have the status of READY"

```
(defun CREATE-ACTIVITY-NODE-FOR-REQUEST (agent request)
  (let (new-task-activity)

    (setf new-task-activity (MAKE-ACTIVITY))
    (setf (ACTIVITY-task                new-task-activity)
          (REQUEST-task-name request))
    (setf (ACTIVITY-task-args           new-task-activity)
          (REQUEST-request-args request))
    (setf (ACTIVITY-parent-task-id      new-task-activity) nil)
    (setf (ACTIVITY-parent-request-id   new-task-activity)
          (REQUEST-request-id request))
    (setf (ACTIVITY-status              new-task-activity) 'READY)
    (setf (REQUEST-request-status request) 'BEING-PROCESSED)

    ; add new node to activity blackboard

    (setf (AGENT-activity-blackboard agent)
          (append (AGENT-activity-blackboard agent)
                  (list new-task-activity)))
  )
)
```

"The following routine processes the activity nodes in an agent's activity blackboard"

```
(defun PROCESS-ACTIVITY-NODES (agent special-processing-constraints)
  (let* ((request-constraints special-processing-constraints)
         (activity-nodes (AGENT-activity-blackboard agent))
         )
    (dolist (a-node activity-nodes)
        (PROCESS-INDIVIDUAL-ACTIVITY-NODE agent a-node request-constraints)
    )
  )
)
```

"The following routine will process a given activity-node that has one or more subtasks as part of its plan (current-task-list). It creates one new activity node for each task in the list and then elaborates the task recursively. Each new activity node is linked to the original activity node and to the originating request."

```
(defun EXPAND-ACTIVITY-NODE (agent activity-node request-constraints)
  (let (sub-task-activity)
```

```
    (dolist (task (ACTIVITY-current-task-list activity-node))
      (setf sub-task-activity (MAKE-ACTIVITY))
      (setf (ACTIVITY-task                sub-task-activity) task)
      (setf (ACTIVITY-status              sub-task-activity) 'READY)
      (setf (ACTIVITY-task-args           sub-task-activity)
                                          (ACTIVITY-task-args activity-node))
      (setf (ACTIVITY-parent-task-id      sub-task-activity)
                                          (ACTIVITY-activity-id activity-node))
      (setf (ACTIVITY-parent-request-id   sub-task-activity)
                                          (ACTIVITY-parent-request-id activity-node))
      (setf (AGENT-activity-blackboard agent)
            (append (AGENT-activity-blackboard agent)
                    (list sub-task-activity)))
    )
    ; now attempt to elaborate the nodes

    (PROCESS-ACTIVITY-NODES agent request-constraints)
  )
)
```

"The following routine processes a single activity node exist in an agent's activity blackboard."

```
(defun PROCESS-INDIVIDUAL-ACTIVITY-NODE (agent activity-node request-constraints)
 (let* (applicable-plans
        target-agent
        current-plan
        task-list
        task-name
        task-parm
       )

   (if (equal (ACTIVITY-status activity-node) 'WORKING)
       ;
       ; If the agent itself is working on it, increment the cycle counter.
       ;
       (INCREMENT-ACTIVITY-CYCLE agent activity-node)

       (if (or (equal (ACTIVITY-status activity-node) 'NO-APPLICABLE-PLANS)
               (equal (ACTIVITY-status activity-node) 'NO-APPLICABLE-PLANS-REPLIED))
           ;
           ; If there is no applicable plans for the task, report to the parent node.
           ;
           (REPORT-TO-PARENT-NODE agent activity-node request-constraints)

           (progn
              ;
              ; If this is the first time in, it tries to find applicable plans for the
              ; task.
              (if (and (equal (ACTIVITY-current-plan-or-task activity-node) nil)
                       (equal (ACTIVITY-current-agent-class  activity-node) nil)
                       (equal (ACTIVITY-current-agent        activity-node) nil))
                  (setf applicable-plans (FIND-APPLICABLE-PLANS
                                          agent
                                          (ACTIVITY-task activity-node)
                                          (ACTIVITY-task-args activity-node)))
              )
              (if (equal applicable-plans nil)
                  ;
                  ; If applicable-plans is empty, its either no applicable plans for the
                  ; task or the task has been sent to other agent to be worked on or the
                  ; agent is working on the task
                  ;
                  (progn
                     (if (equal (ACTIVITY-status activity-node)
                                'AWAITING-SUBTASK-DISTRIBUTION)
                         ;
                         ; The task has been sent to other agents. Here invokes
                         ; CHECK-CHILDREN ACTIVITY-STATUS to see if the task is done or if
```

```
            ; there is any problem.
            ;
            (CHECK-CHILDREN-ACTIVITY-STATUS agent activity-node)

            (progn
               ;
               ; Here tries to find suitable agent to work on the task and
               ; checks to see if the task has any problem.
               ;
               (setf target-agent (FIND-SUITABLE-AGENT agent activity-node
                                   request-constraints))
               (setf task-name    (ACTIVITY-task activity-node))
               (if (and (equal target-agent nil)
                        (not (equal (ACTIVITY-status activity-node)
                                    'REQUEST-SENT))
                        (not (equal (ACTIVITY-status activity-node)
                                    'NO-PROBLEM))
                        (not (equal (ACTIVITY-status activity-node)
                                    'NO-PROBLEM-REPLIED))
                        (not (equal (ACTIVITY-status activity-node)
                                    'WORKING)))
                   (progn
                      (format t "    Set task: ~A status to
                                     NO APPLICABLE PLANS"
                              (ACTIVITY-task activity-node))
                      (terpri)
                      (setf (ACTIVITY-status activity-node)
                            'NO-APPLICABLE-PLANS)
                   )
                   (if (not (equal target-agent nil))
                       (progn
                          ;
                          ; After a suitable agent is found, sends off the
                          ; task to such agent.
                          ;
                          (setf (ACTIVITY-status activity-node)
                                'REQUEST-SENT)
                          (SEND-REQUEST (AGENT-agent-name agent)
                                        target-agent
                                        'task
                                        task-name
                                        task-parm
                                       (ACTIVITY-activity-id activity-node))
                       )
                   )
               )
            )
         )
      )
   )
)
(progn
   ;
   ; The applicable plans list is not empty.
   ;
   (if (equal (ACTIVITY-status activity-node) 'READY)
       (progn
          ;
          ; If the activity node is newly created, i.e. status is READY,
          ; it chooses the first plan from the applicable plans to start
          ; work on the task. Constraints can be implemented here to
          ; assist better selection in choosing a plan.
          ;
          (setf current-plan     (FIRST applicable-plans))
          (setf applicable-plans (REST  applicable-plans))
          (setf task-list
                (AND current-plan
                     ; nil task-list if TASK structure instead of PLAN
                     (not (TASK-p current-plan))
                     (PLAN-task-list current-plan)))

          (setf (ACTIVITY-applicable-plans-and-tasks activity-node)
                applicable-plans)
          (setf (ACTIVITY-current-plan-or-task       activity-node)
                current-plan)
          (setf (ACTIVITY-current-task-list          activity-node)
                task-list)
```

```
                                (if (null task-list)
                                    (if (equal (TASK-agent-class-list current-plan) nil)
                                        ;
                                        ; If the task's agent-class-list is nil, it implies
                                        ; that the task can be done by the agent itself
                                        ; without distributing to other agents. Here it
                                        ; increments the cycle counter from default value of
                                        ; 0 to 1
                                        (progn
                                           (setf (ACTIVITY-status activity-node) 'WORKING)
                                           (INCREMENT-ACTIVITY-CYCLE agent activity-node)
                                        )
                                        (progn
                                           (format t "    Set task: ~A status to
                                                          AWAITING DISTRIBUTION"
                                                 (ACTIVITY-task activity-node))
                                           (terpri)
                                           (setf (ACTIVITY-status activity-node)
                                                 'AWAITING-DISTRIBUTION)
                                        )
                                    )
                                    (progn
                                       (if (equal (ACTIVITY-parent-task-id activity-node) nil)
                                           (progn
                                              (format t "    Set task: ~A status to
                                                             AWAITING SUBTASK DISTRIBUTION"
                                                    (ACTIVITY-task        activity-node))
                                              (terpri)
                                              (setf (ACTIVITY-status activity-node)
                                                    'AWAITING-SUBTASK-DISTRIBUTION)
                                           )
                                       )
                                       ;
                                       ; Here expands the activity node based on the chosen
                                       ; plan.
                                       (EXPAND-ACTIVITY-NODE agent activity-node
                                                             request-constraints)
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)
```

"The following routine increments the cycle counter of an activity node by one. If the cycle counter reaches the value set in the task constraints list, the status will be set to NO-PROBLEM."

```
(defun INCREMENT-ACTIVITY-CYCLE (agent activity-node)
   (let ((task-constraints nil)
         (task-cycle       nil))
       (progn
           (dolist (a-task (AGENT-task-list agent))
               (if (equal (TASK-task-name a-task) (ACTIVITY-task activity-node))
                   (setf task-constraints (TASK-task-constraints a-task))
               )
           )
           (setf task-cycle (cadr (assoc 'CYCLE task-constraints)))
           (if (equal (ACTIVITY-cycle activity-node) task-cycle)
               (progn
                   (format t "    Set task: ~A status to NO PROBLEM"
                       (ACTIVITY-task activity-node))
                   (terpri)
                   (setf (ACTIVITY-status activity-node) 'NO-PROBLEM)
```

```
                )
                (progn
                    (setf (ACTIVITY-cycle activity-node)
                          (+ (ACTIVITY-cycle activity-node) 1))
                    (format t "    Set task: ~A cycle to ~A (Max = ~A)"
                        (ACTIVITY-task  activity-node)
                        (ACTIVITY-cycle activity-node)
                        task-cycle)
                    (terpri)
                )
            )
        )
    )
)
```

"The following routine checks all the children activity nodes (if there exist any) of a given activity node of an agent. If all the children activity nodes' statuses are NO-PROBLEM, then the status of the given activity node will be set to NO-PROBLEM too."

```
(defun CHECK-CHILDREN-ACTIVITY-STATUS (agent activity-node)
  (let ((activity-id (ACTIVITY-activity-id activity-node))
        (no-problem 'T))

    (dolist (a-node (AGENT-activity-blackboard agent))
      (if (and (equal (ACTIVITY-parent-task-id a-node) activity-id)
               (not (equal (ACTIVITY-status a-node) 'NO-PROBLEM))
               (not (equal (ACTIVITY-status a-node) 'NO-PROBLEM-REPLIED)))
          (setf no-problem nil)
      )
    )
    (if (equal no-problem 'T)
        (progn
           (format t "    Set task: ~A status to NO PROBLEM"
                 (ACTIVITY-task activity-node))
           (terpri)
           (setf (ACTIVITY-status activity-node) 'NO-PROBLEM)
        )
    )
  )
)
```

"The following routine will either send a task to a suitable agent or return nil to indicate no suitable agent can be found. Here we just choose the first agent to be the suitable agent. Constraints can be implemented later to assist in finding suitable agent."

```
(defun FIND-SUITABLE-AGENT (agent activity-node request-constraints)
  (let* ((task-name (ACTIVITY-task activity-node))
         (target-agent-classes nil)
         (target-agent-class   nil)
         (target-agents        nil)
         (target-agent         nil)
        )

    (if (and (equal (ACTIVITY-current-agent-class activity-node) nil)
             (equal (ACTIVITY-current-agent       activity-node) nil)
             (not (equal (ACTIVITY-status         activity-node)
                         'NO-APPLICABLE-PLANS))
             (not (equal (ACTIVITY-status         activity-node)
                         'NO-APPLICABLE-PLANS-REPLIED))
             (not (equal (ACTIVITY-status         activity-node) 'TASK-PROBLEM)))
        (progn
          (setf target-agent-classes (FIND-AGENT-CLASS-LIST-FOR-TASK agent task-name))
          (setf target-agent-class   (first target-agent-classes))
```

```
            (setf target-agent-classes (rest  target-agent-classes))
            (setf (ACTIVITY-applicable-agent-classes activity-node) target-agent-classes)
            (setf (ACTIVITY-current-agent-class      activity-node) target-agent-class)
            (setf target-agents (FIND-AGENT-LIST-FOR-AGENT-CLASS agent target-agent-class))
            (setf target-agent  (first target-agents))
            (setf target-agents (rest  target-agents))
            (setf (ACTIVITY-applicable-agents activity-node) target-agents)
            (setf (ACTIVITY-current-agent     activity-node) target-agent)
          )
          (progn
            (if (or (and (equal (ACTIVITY-current-agent activity-node) nil)
                         (not (equal (ACTIVITY-status activity-node)
                                     'NO-APPLICABLE-PLANS))
                         (not (equal (ACTIVITY-status activity-node)
                                     'NO-APPLICABLE-PLANS-REPLIED)))
                    (and (equal (ACTIVITY-applicable-agents activity-node) nil)
                         (equal (ACTIVITY-status activity-node) 'TASK-PROBLEM)))
                (progn
                  (setf target-agent-classes (ACTIVITY-applicable-agent-classes
                                              activity-node))
                  (setf target-agent-class   (first target-agent-classes))
                  (setf target-agent-classes (rest  target-agent-classes))
                  (setf (ACTIVITY-applicable-agent-classes activity-node)
                        target-agent-classes)
                  (setf (ACTIVITY-current-agent-class      activity-node)
                        target-agent-class)
                  (setf target-agents (FIND-AGENT-LIST-FOR-AGENT-CLASS agent
                        target-agent-class))
                  (setf target-agent  (first target-agents))
                  (setf target-agents (rest  target-agents))
                  (setf (ACTIVITY-applicable-agents activity-node) target-agents)
                  (setf (ACTIVITY-current-agent     activity-node) target-agent)
                )
                (progn
                  (if (or (equal (ACTIVITY-status activity-node) 'READY)
                          (equal (ACTIVITY-status activity-node) 'AWAITING-DISTRIBUTION)
                          (equal (ACTIVITY-status activity-node) 'TASK-PROBLEM))
                      (progn
                        (setf target-agents (ACTIVITY-applicable-agents activity-node))
                        (setf target-agent  (first target-agents))
                        (setf target-agents (rest  target-agents))
                        (setf (ACTIVITY-applicable-agents activity-node) target-agents)
                        (setf (ACTIVITY-current-agent     activity-node) target-agent)
                      )
                  )
                )
            )
          )
        )
        target-agent
      )
)
```

"The following routine will either set the parent node of the input activity to have status of TASK-PROBLEM or if the activity is requested from another agent, a notification will be sent back to the request agent with TASK-PROBLEM message."

```
(defun REPORT-TO-PARENT-NODE (agent activity request-constraints)
   (let ((req nil)
         (current-plan nil)
         (applicable-plans nil)
         (task-list nil))

      (if (not (equal (ACTIVITY-status activity) 'NO-APPLICABLE-PLANS-REPLIED))
          (if (equal (ACTIVITY-parent-task-id activity) nil)
              (progn
                  ;
                  ; If the parent task id of the activity node is nil, it implies that
                  ; the task is requested from another agent.
                  ;
```

```
                    (if (equal (ACTIVITY-applicable-plans-and-tasks activity) nil)
                        (progn
                            ;
                            ; If there is no other applicable plans available, it sends a
                            ; notif back to the request agent indicating there is a
                            ; problem.
                            (setf req (FIND-REQUEST-GIVEN-ID agent
                                      (ACTIVITY-parent-request-id activity)))
                            (if (not (equal (REQUEST-from-agent req) nil))
                                (progn
                                    (SEND-NOTIFICATION (REQUEST-to-agent req)
                                                       (REQUEST-from-agent req)
                                                       (REQUEST-request-id req)
                                                       (REQUEST-task-name req)
                                                       'TASK-PROBLEM)
                                    (setf (ACTIVITY-status activity)
                                          'NO-APPLICABLE-PLANS-REPLIED)
                                )
                            )
                        )
                        (progn
                            ;
                            ; If alternative plan is available, it will try to use it.
                            ;
                            (setf (ACTIVITY-status activity)
                                  'AWAITING-SUBTASK-DISTRIBUTION)
                            (setf applicable-plans
                                  (ACTIVITY-applicable-plans-and-tasks activity))
                            (setf current-plan      (first applicable-plans))
                            (setf applicable-plans  (rest  applicable-plans))
                            (setf (ACTIVITY-current-plan-or-task       activity)
                                  current-plan)
                            (setf (ACTIVITY-applicable-plans-and-tasks activity)
                                  applicable-plans)
                            (setf task-list
                                  (AND current-plan
                                       ; nil task-list if TASK structure instead of PLAN
                                       (not (TASK-p current-plan))
                                       (PLAN-task-list current-plan)))
                            (setf (ACTIVITY-current-task-list activity) task-list)
                            (EXPAND-ACTIVITY-NODE agent activity request-constraints)
                        )
                    )
                )
                ;
                ; Here it deletes all the nodes created for the previous plan which has
                ; been failed.
                ;
                (dolist (a-node (AGENT-activity-blackboard agent))
                    (if (equal (ACTIVITY-activity-id a-node)
                               (ACTIVITY-parent-task-id activity))
                        (progn
                            (format t "    Set task: ~A status to TASK PROBLEM"
                                  (ACTIVITY-task a-node))
                            (terpri)
                            (setf (ACTIVITY-status a-node)   'TASK-PROBLEM)
                            (setf (ACTIVITY-status activity) 'NO-APPLICABLE-PLANS-REPLIED)
                            (DELETE-ACTIVITY-SUB-TASK-NODES agent activity)
                        )
                    )
                )
            )
        )
    )
)
```

"The following routine will recreate the agent's activity blackboard by deleting any tasks that have the given activity node as its parent."

```
(defun DELETE-ACTIVITY-SUB-TASK-NODES (agent activity)
  (let ((new-bb nil)
        (activity-id (ACTIVITY-activity-id activity)))

    (dolist (node (AGENT-activity-blackboard agent))
      (if (equal (ACTIVITY-parent-task-id node) activity-id)
          (progn
              (if (not (equal (ACTIVITY-current-agent node) nil))
                  (SEND-REQUEST (AGENT-agent-name agent)
                                (ACTIVITY-current-agent node)
                                'CLEAN-UP
                                (ACTIVITY-task node)
                                nil
                                (ACTIVITY-activity-id node))
              )
          )
          (setf new-bb (append new-bb (list node)))
      )
    )
    (setf (AGENT-activity-blackboard agent) new-bb)
    ;
    ; Delete the activity node itself
    ;
    (setf new-bb nil)
    (dolist (node (AGENT-activity-blackboard agent))
       (if (not (equal (ACTIVITY-activity-id node) activity-id))
           (setf new-bb (append new-bb (list node)))
       )
    )
    (setf (AGENT-activity-blackboard agent) new-bb)
  )
)
```

"This routine will search an agent's plan list to find zero or more plans that can be applied to a given request. For now we select all the plans on the agent's plan list that match the task associated with the request (no other details such as task arguments are considered at present)."

```
(defun FIND-APPLICABLE-PLANS (agent task-name task-arguments)
  (let ((applicable-plans nil))
    ; first find all possible task knowledge sources that match the task
    ; These represent alternatives that can be used to directly distribute
    ; the task
    (dolist (tk (AGENT-task-list agent))
      (if (equal (TASK-task-name tk) task-name)
        (setf applicable-plans
              (append applicable-plans
                      (list tk))))
      )

    (dolist (p (AGENT-plan-list agent))
      (if (equal (PLAN-task-name p) task-name)
        (setf applicable-plans
              (append applicable-plans (list p))))
      )
    applicable-plans)
)
```

"The following routine will return a list of the ACTIVITY-id(s) associated with a given task in a given agent.

```
(defun FIND-ACTVITY-ID (agent-name task-name)
  (let ((agent (FIND-AGENT agent-name))
        (task-ids nil))

    (dolist (a-node (AGENT-activity-blackboard agent))
      (if (equal (ACTIVITY-task a-node) task-name)
          (setf task-ids
                (append task-ids (list (ACTIVITY-activity-id a-node))))
      )
    )
    task-ids)
)
```

"The following routine will search a given agent's task knowledge to find and return a list of agent classes (possibly nil) that are known by the agent to be able to process the given task-name.

```
(defun FIND-AGENT-CLASS-LIST-FOR-TASK (agent task-name)
  (let* ((task-list  (AGENT-task-list agent))
         (class-lists nil))

    (dolist (x task-list)
      (if (equal (TASK-task-name x) task-name)
          (setf class-lists
                (append class-lists (TASK-agent-class-list x)))
      )
    )
    class-lists
  )
)
```

"The following routine finds a list of actual agents that belong to the given class of agents. Search is based on the information local to the given agent."

```
(defun FIND-AGENT-LIST-FOR-AGENT-CLASS (agent agent-class)
  (let* ((agent-lists nil))
    (dolist (a-class (AGENT-agent-class-list agent))
      (if (equal (AGENT-CLASS-class-name a-class) agent-class)
          (setf agent-lists
                (append agent-lists (AGENT-CLASS-agent-list a-class)))
      )
    )
    agent-lists
  )
)
```

"The following routine processes the incoming and outgoing notification queue of an agent."

```
(defun PROCESS-NOTIFICATIONS (agent special-processing-constraints)
  (let* ((process-constraints
              (OR special-processing-constraints
                  (AGENT-processing-constraints agent)))
         (request-constraints
                   (PROCESSING-CONSTRAINTS-request-constraints process-constraints))
         )
```

```
      (progn
         (PROCESS-INCOMING-NOTIFICATIONS agent request-constraints)
         (PROCESS-OUTGOING-NOTIFICATIONS agent)
      )
   )
)
```

"The following routine processes all the incoming notifications in the incoming notification queue. If the incoming notification is a new one and the notification info states that it has no problem in working on the request, then the corresponding activity id status will be set to NO-PROBLEM. If the incoming notification indicates there is a task problem, the corresponding activity id status will be set to TASK-PROBLEM and a clean-up request will be sent to the failing agent to clean-up the activity nodes. "

```
(defun PROCESS-INCOMING-NOTIFICATIONS (agent request-constraints)
 (let* ((activity-node nil)
        (temp-request  nil))
   (dolist (i-notif (AGENT-incoming-notification-list agent))
      (if (and (equal (NOTIFICATION-notif-status i-notif) 'JUST-RECEIVED)
               (equal (NOTIFICATION-notif-info   i-notif) 'NO-PROBLEM))
          (progn
              (setf (NOTIFICATION-notif-status i-notif) 'PROCESSED)
              (setf activity-node (FIND-ACTIVITY-BY-REQ-ID agent
                                    (NOTIFICATION-request-id i-notif)))
              (if (not (equal activity-node nil))
                  (progn
                      (setf (ACTIVITY-status activity-node) 'NO-PROBLEM)
                      (format t "    Set task: ~A status to NO PROBLEM"
                            (ACTIVITY-task activity-node))
                      (terpri)
                  )
              )
          )
      )
      (if (and (equal (NOTIFICATION-notif-status i-notif) 'JUST-RECEIVED)
               (equal (NOTIFICATION-notif-info   i-notif) 'TASK-PROBLEM))
          (progn
              (setf (NOTIFICATION-notif-status i-notif) 'PROCESSED)
              (setf activity-node (FIND-ACTIVITY-BY-REQ-ID agent
                                    (NOTIFICATION-request-id i-notif)))
              (if (not (equal activity-node nil))
                  (progn
                      (setf (ACTIVITY-status activity-node) 'TASK-PROBLEM)
                      (format t "    Set task: ~A status to TASK PROBLEM"
                            (ACTIVITY-task activity-node))
                      (terpri)
                      (SEND-REQUEST (AGENT-agent-name agent)
                                    (ACTIVITY-current-agent activity-node)
                                    'CLEAN-UP
                                    (ACTIVITY-task activity-node)
                                    nil
                                    (ACTIVITY-activity-id activity-node))
                  )
              )
              (if (equal (AGENT-agent-name agent) 'USER)
                  (progn
                      (setf temp-request (first (AGENT-outgoing-request-list agent)))
                      (SEND-REQUEST (AGENT-agent-name agent)
                                    'Mark-House-Building
                                    'CLEAN-UP
                                    'House-Building
                                    nil
                                    (REQUEST-request-id temp-request))
                  )
              )
          )
      )
   )
)
```

```
  )
 )
```

"The following routine will return the activity node associated with a given notification id in a given agent. The notification id should be equal to one of the activity id in the activity blackboard."

```
(defun FIND-ACTIVITY-BY-REQ-ID (agent req-id)
  (let ((activity-node nil))

    (dolist (a-node (AGENT-activity-blackboard agent))
      (if (equal (ACTIVITY-activity-id a-node) req-id)
          (setf activity-node a-node)
      )
    )
    activity-node
  )
)
```

"The following routine generates notification messages to request agents when the status of the request shows NO-PROBLEM."

```
(defun PROCESS-OUTGOING-NOTIFICATIONS (agent)
   (let* ((root-nodes (FIND-ROOT-ACTIVITY-NODES agent))
          (req nil))

      (dolist (a-node root-nodes)
         (if (equal (ACTIVITY-status a-node) 'NO-PROBLEM)
             (progn
                 (setf (ACTIVITY-status a-node) 'NO-PROBLEM-REPLIED)
                 (setf req (FIND-REQUEST-GIVEN-ID agent
                           (ACTIVITY-parent-request-id a-node)))
                 (if (not (equal (REQUEST-from-agent req) nil))
                     (SEND-NOTIFICATION (REQUEST-to-agent req) (REQUEST-from-agent req)
                                        (REQUEST-request-id req) (REQUEST-task-name req)
                                        'NO-PROBLEM)
                 )
             )
         )
      )
   )
)
```

"The following routine finds all the root activity nodes of an agent. A root activity node is an activity node created as a result of an incoming request from another agent."

```
(defun FIND-ROOT-ACTIVITY-NODES (agent)
   (let* ((root-nodes  nil))

      (dolist (a-node (AGENT-activity-blackboard agent))
          (if (equal (ACTIVITY-parent-task-id a-node) nil)
              (setf root-nodes (append root-nodes (list a-node)))
          )
      )
      root-nodes
   )
)
```

"The following routine will print out all of the information related to a given agent."

```
(defun PRINT-AGENT-INFO (agent-name)
  (let ((agent (FIND-AGENT agent-name)))

    (format t "AGENT: ~A" (AGENT-agent-name agent))
    (terpri)
    (format t "   Plan:")
    (terpri)
    (PRINT-PLAN-LIST              agent)
    (terpri)
    (format t "   Task:")
    (terpri)
    (PRINT-TASK-LIST              agent)
    (terpri)
    (format t "   Agent Class:")
    (terpri)
    (PRINT-AGENT-CLASS-LIST       agent)
    (terpri)
    (PRINT-CONSTRAINT-LIST        agent)
    (PRINT-INCOMING-REQUESTS      agent)
    (PRINT-OUTGOING-REQUESTS      agent)
    (PRINT-INCOMING-NOTIFICATIONS agent)
    (PRINT-OUTGOING-NOTIFICATIONS agent)
    (PRINT-ACTIVITY-BLACKBOARD    agent)
    )
)
```

"The following routine will print out all the plan information known to a given agent."

```
(defun PRINT-PLAN-LIST (agent)
  (dolist (p (AGENT-plan-list agent))
    (format t "      Task: ~A"
            (PLAN-task-name p))
    (terpri)
    (princ    "         Sub-Tasks: ")
    (terpri)
    (dolist (task (PLAN-task-list p))
      (princ "                  ")
      (princ task)
      (terpri)
      )
    (terpri)
    )
)
```

"The following routine will print out all of the task information known to a given agent."

```
(defun PRINT-TASK-LIST (agent)
  (dolist (task (AGENT-task-list agent))
    (format t "      Task:  ~A"
          (TASK-task-name task))
    (terpri)
    (format t "         Agent Classes:  ~A"
          (TASK-agent-class-list task))
    (terpri)
    )
)
```

"The following routine will print a list of agents for each agent class known to a given agent."

```
(defun PRINT-AGENT-CLASS-LIST (agent)
  (dolist (agent-class (AGENT-agent-class-list agent))
    (format t "      Class Name:  ~A"
            (AGENT-CLASS-class-name agent-class))
    (terpri)
    (format t "                   Agents:  ~A"
            (AGENT-CLASS-agent-list agent-class))
    (terpri)
    )
)
```

"The following routine will print the constraints associated with a given agent."

```
(defun PRINT-CONSTRAINT-LIST (agent)
  (format t "   Constraints:")
  (terpri)
  (dolist (r (AGENT-constraint-list agent))
    (format t "         ~A" r)
    (terpri)
  )
  (terpri)
)
```

"The following routine will print the requests that have been received by the given agent."

```
(defun PRINT-INCOMING-REQUESTS (agent)
  (format t   "   Incoming Requests:")
  (terpri)
  (dolist (r (AGENT-incoming-request-list agent))
    (format t "        ~A:  ~A  (~A)"
            (REQUEST-from-agent   r)
            (REQUEST-request-type r)
            (REQUEST-task-name    r))
    (terpri)
  )
)
```

"The following routine will print the requests that have been sent out by the given agent."

```
(defun PRINT-OUTGOING-REQUESTS (agent)
  (format t  "   Outgoing Requests:")
  (terpri)
  (dolist (r (AGENT-outgoing-request-list agent))
    (format t "        ~A:  ~A  (~A)"
            (REQUEST-to-agent   r)
            (REQUEST-request-type r)
            (REQUEST-task-name    r))
  (terpri)
  )
)
```

"The following routine will print the notifications that have been received by the given agent."

```
(defun PRINT-INCOMING-NOTIFICATIONS (agent)
  (format t "   Incoming Notification:")
  (terpri)
  (dolist (n (AGENT-incoming-notification-list agent))
    (format t "        ~A:  ~A  (~A)"
            (NOTIFICATION-from-agent   n)
            (NOTIFICATION-notif-type   n)
            (NOTIFICATION-notif-info   n))
    (terpri)
  )
)
```

"The following routine will print the notifications that have been sent out by the given agent."

```
(defun PRINT-OUTGOING-NOTIFICATIONS (agent)
  (format t "   Outgoing Notification:")
  (terpri)
  (dolist (n (AGENT-outgoing-notification-list agent))
    (format t "        ~A:  ~A  (~A)"
            (NOTIFICATION-to-agent     n)
            (NOTIFICATION-notif-type   n)
            (NOTIFICATION-notif-info   n))
    (terpri)
  )
)
```

"The following routine will print the activity node information of the activity blackboard."

```
(defun PRINT-ACTIVITY-BLACKBOARD (agent)
  (format t "   Activity Blackboard:")
  (terpri)
  (dolist (node (AGENT-activity-blackboard agent))
    (format t "     Task: ~A  Status: ~A  Current Agent: ~A"
            (ACTIVITY-task          node)
            (ACTIVITY-status        node)
            (ACTIVITY-current-agent node))
    (terpri)
    )
)
```

"The following routine initiates the first request."

```
(defun initiate-request ()
   (send-request 'user 'Mark-House-Building 'task 'House-Building nil nil)
)
```

"The following routines define the menu bar and dialog boxes."

```
(defvar commands-menu)

(defun start-up-routine ()
  (progn
     (if (not (find-menu "Commands"))
         (progn
```

```
                (setq commands-menu
                     (Make-Instance 'Menu
                     :Menu-Title    "Commands"
                     :Menu-Items
                       (List
                              (Make-Instance 'Menu-Item
                                :Menu-Item-Title
                                "Choose A Dataset......"
                                :Command-Key
                                #\C
                                :Menu-Item-Action
                                #'(Lambda Nil (Create-Dataset-Dialog)))
                              (Make-Instance 'Menu-Item
                                :Menu-Item-Title
                                "Initiate Request"
                                :Command-Key
                                #\R
                                :Menu-Item-Action
                                #'(Lambda Nil (Initiate-Request)))
                              (Make-Instance 'Menu-Item
                                :Menu-Item-Title
                                "Process Agents"
                                :Command-Key
                                #\P
                                :Menu-Item-Action
                                #'(Lambda Nil (Process-Agents)))
                              (Make-Instance 'Menu-Item
                                :Menu-Item-Title
                                "Print Agent Info....."
                                :Command-Key
                                #\I
                                :Menu-Item-Action
                                #'(Lambda Nil (Create-Print-Dialog)))
                              (Make-Instance 'Menu-Item
                                :Menu-Item-Title
                                "Help"
                                :Command-Key
                                #\H
                                :Menu-Item-Action
                                #'(Lambda Nil (Help)))
                       ))
                )
                (menu-install commands-menu)
           )
        )
     )
)


(defun Create-dataset-dialog ()

 (Make-Instance 'Dialog
  :Window-Type
  :Document
  :Window-Title
  "Choose A Dataset"
  :View-Position
  ':Centered
  :View-Size
  #@(300 150)
  :View-Font
  '("Chicago" 12 :Srcor :Plain)
  :View-Subviews
  (List
        (Make-Dialog-Item
          'Button-Dialog-Item
          #@(10 13)
          #@(78 16)
          "Dataset A"
          #'(Lambda (Item) Item (Define-Dataset-A))
          :Default-Button
          Nil)
        (Make-Dialog-Item
          'Button-Dialog-Item
          #@(100 13)
```

```
            #@(78 16)
            "Dataset B1"
            #'(Lambda (Item) Item (Define-Dataset-B-1))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(100 32)
            #@(78 16)
            "Dataset B2"
            #'(Lambda (Item) Item (Define-Dataset-B-2))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(190 13)
            #@(78 16)
            "Dataset C1"
            #'(Lambda (Item) Item (Define-Dataset-C-1))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(190 32)
            #@(78 16)
            "Dataset C2"
            #'(Lambda (Item) Item (Define-Dataset-C-2))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(10 57)
            #@(78 16)
            "Dataset D1"
            #'(Lambda (Item) Item (Define-Dataset-D-1))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(10 77)
            #@(78 16)
            "Dataset D2"
            #'(Lambda (Item) Item (Define-Dataset-D-2))
            :Default-Button
            Nil)
 ))
)


(defun Create-Print-Dialog ()

 (Make-Instance 'Dialog
  :Window-Type
  :Document
  :Window-Title
  "Print Agent Info"
  :View-Position
  ':Centered
  :View-Size
  #@(300 150)
  :View-Font
  '("Chicago" 12 :Srcor :Plain)
  :View-Subviews
  (List
        (Make-Dialog-Item
          'Button-Dialog-Item
          #@(10 13)
          #@(78 16)
          "Agent 1"
          #'(Lambda (Item) Item (Print-Agent-Info 'Mark-House-Building))
          :Default-Button
          Nil)
        (Make-Dialog-Item
          'Button-Dialog-Item
          #@(100 13)
```

```
            #@(78 16)
            "Agent 2"
            #'(Lambda (Item) Item (Print-Agent-Info 'John-House-Building))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(190 13)
            #@(78 16)
            "Agent 3"
            #'(Lambda (Item) Item (Print-Agent-Info 'Tom-House-Building))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(10 32)
            #@(78 16)
            "Agent 4"
            #'(Lambda (Item) Item (Print-Agent-Info 'Paul-House-Building))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(100 32)
            #@(78 16)
            "Agent 5"
            #'(Lambda (Item) Item (Print-Agent-Info 'Doug-House-Building))
            :Default-Button
            Nil)
          (Make-Dialog-Item
            'Button-Dialog-Item
            #@(190 32)
            #@(78 16)
            "Agent 6"
            #'(Lambda (Item) Item (Print-Agent-Info 'Rick-House-Building))
            :Default-Button
            Nil)
 ))
)


(defun help ()
   (terpri)
   (format t "    ")
   (terpri)
   (format t "Basic steps to run the application:")
   (terpri)
   (terpri)
   (format t "- Choose A Dataset")
   (terpri)
   (format t "- Initiate Request")
   (terpri)
   (format t "- Process Agents")
   (terpri)
   (terpri)
   (format t "Dataset A represents a smooth run; i.e. no change in agent, class or plan.")
   (terpri)
   (terpri)
   (format t "Dataset B1 represents a change in agent when one fails.")
   (terpri)
   (format t "       Tom-House-Building fails to perform Electricity for
            John-House-Building, John-House-Building then send the same task to
            Paul-House-Building")
   (terpri)
   (format t "       based on the same agent class. Paul-House-Building has no problem
            in doing the task.")
   (terpri)
   (format t "Dataset B2 similar to b-1 except that Paul-House-Building fails to
            perform the task.")
   (terpri)
   (terpri)
   (format t "Dataset C1 represents a change in agent class when one fails.")
   (terpri)
   (format t "       Tom-House-Building fails to perform Electricity for
            John-House-Building, John-House-Building then send the same task to
```

```
            Paul-House-Building")
    (terpri)
    (format t "       based on different agent class. Paul-House-Building has no problem
            in doing the task.")
    (terpri)
    (format t "Dataset C2 similar to c-1 except that Paul-House-Building fails to
            perform the task.")
    (terpri)
    (terpri)
    (format t "Dataset D1 represents a change in agent plan when one fails.")
    (terpri)
    (format t "       Tom-House-Building fails to perform Plumbing for John-House-Building,
            John-House-Building then uses another plan to")
    (terpri)
    (format t "       send Electricity to Tom-House-Building. Tom-House-Building has
            no problem in doing the task.")
    (terpri)
    (format t "Dataset D2 similar to d-1 except that Tom-House-Building still has problem
            in doing the task.")
    (terpri)
    (terpri)
    (format t "Note that some responses may take more than one cycle of (process-agents).")
    (terpri)
    (format t "When you see no response in one cycle, some internal function might be
            going on.")
    (terpri)
    (terpri)
    (format t "Choose Help at anytime to show the above notes.")
)

(start-up-routine)

(help)
```

Appendix C : Test Data and Sample Runs

Test Data:

```
(defvar user)
(defvar agent-1)
(defvar agent-2)
(defvar agent-3)
(defvar agent-4)
(defvar agent-5)
(defvar agent-6)


(defun setup-user ()
   (setq user (MAKE-AGENT :agent-name 'USER))
)


(defun setup-agent1-a ()
   (setq agent-1 (MAKE-AGENT :agent-name 'Mark-House-Building
                     :plan-list (LIST
                                 (MAKE-PLAN :plan-name 'Plan-Mark-House-Building
                                            :task-name 'House-Building
                                            :task-list '(Build-Exterior Build-Interior))
                                )
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Build-Exterior
                                            :agent-class-list
                                             '(Mark-Class-Build-Exterior))
                                 (MAKE-TASK :task-name 'Build-Interior
                                            :agent-class-list
                                             '(Mark-Class-Build-Interior))
                                )
                     :agent-class-list
                                 (LIST
                                  (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Exterior
                                                    :agent-list '(John-House-Building))
                                  (MAKE-AGENT-CLASS :class-name 'Mark-Class-Build-Interior
                                                    :agent-list '(John-House-Building)))
                     )
      )
)


(defun setup-agent2-a ()
   (setq agent-2 (MAKE-AGENT :agent-name 'John-House-Building
                     :plan-list (LIST
                                 (MAKE-PLAN :plan-name 'PLAN-John-House-Building-A
                                            :task-name 'Build-Exterior
                                            :task-list
                                             '(Lay-Foundation Build-House-Frame))
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Interior
                                            :task-name 'Build-Interior
                                            :task-list '(Plumbing Electricity Decoration))
                                )
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Lay-Foundation
                                            :agent-class-list
                                             '(John-Class-Lay-Foundation))
                                 (MAKE-TASK :task-name 'Build-House-Frame
                                            :agent-class-list
                                             '(John-Class-Build-House-Frame))
                                 (MAKE-TASK :task-name 'Plumbing
                                            :agent-class-list '(John-Class-Plumbing))
                                 (MAKE-TASK :task-name 'Electricity
                                            :agent-class-list '(John-Class-Electricity))
                                 (MAKE-TASK :task-name 'Decoration
                                            :task-constraints '((CYCLE 24))
                                            :agent-class-list nil)
                                )
                     :agent-class-list
```

```
                                    (LIST
                                     (MAKE-AGENT-CLASS :class-name
                                                 'John-Class-Lay-Foundation
                                                       :agent-list
                                                 '(Tom-House-Building Paul-House-Building))
                                     (MAKE-AGENT-CLASS :class-name
                                                 'John-Class-Build-House-Frame
                                                       :agent-list '(Paul-House-Building))
                                     (MAKE-AGENT-CLASS :class-name
                                                 'John-Class-Plumbing
                                                       :agent-list
                                                 '(Tom-House-Building Paul-House-Building))
                                     (MAKE-AGENT-CLASS :class-name
                                                 'John-Class-Electricity
                                                       :agent-list
                                                 '(Tom-House-Building Paul-House-Building))
                                     (MAKE-AGENT-CLASS :class-name
                                                 'John-Class-Decoration
                                                       :agent-list
                                                 '(Tom-House-Building Paul-House-Building)))
                        )
      )
)


(defun setup-agent2-b ()
   (setq agent-2 (MAKE-AGENT :agent-name 'John-House-Building
                      :plan-list (LIST
                                  (MAKE-PLAN :plan-name 'Plan-John-Build-Exterior
                                             :task-name 'Build-Exterior
                                             :task-list
                                              '(Lay-Foundation Build-House-Frame))
                                  (MAKE-PLAN :plan-name 'Plan-John-Build-Interior
                                             :task-name 'Build-Interior
                                             :task-list '(Plumbing Electricity Decoration))
                                  )
                     :task-list (LIST
                                  (MAKE-TASK :task-name 'Lay-Foundation
                                             :agent-class-list
                                              '(John-Class-Lay-Foundation))
                                  (MAKE-TASK :task-name 'Build-House-Frame
                                             :agent-class-list
                                              '(John-Class-Build-House-Frame))
                                  (MAKE-TASK :task-name 'Plumbing
                                             :agent-class-list '(John-Class-Plumbing))
                                  (MAKE-TASK :task-name 'Electricity
                                             :agent-class-list
                                   '(John-Class-Electricity-A John-Class-Electricity-B))
                                  (MAKE-TASK :task-name 'Decoration
                                             :task-constraints '((CYCLE 24))
                                             :agent-class-list nil)
                                  )
                     :agent-class-list
                                  (LIST
                                   (MAKE-AGENT-CLASS :class-name 'John-Class-Lay-Foundation
                                                     :agent-list
                                                '(Tom-House-Building Paul-House-Building))
                                   (MAKE-AGENT-CLASS :class-name
                                                'John-Class-Build-House-Frame
                                                     :agent-list '(Paul-House-Building))
                                   (MAKE-AGENT-CLASS :class-name 'John-Class-Plumbing
                                                     :agent-list
                                                '(Tom-House-Building Paul-House-Building))
                                   (MAKE-AGENT-CLASS :class-name 'John-Class-Electricity-A
                                                     :agent-list '(Tom-House-Building))
                                   (MAKE-AGENT-CLASS :class-name 'John-Class-Electricity-B
                                                     :agent-list '(Paul-House-Building))
                                   (MAKE-AGENT-CLASS :class-name 'John-Class-Decoration
                                                     :agent-list
                                                '(Tom-House-Building Paul-House-Building)))
                     )
      )
)
```

```
(defun setup-agent2-c ()
   (setq agent-2 (MAKE-AGENT :agent-name 'John-House-Building
                     :plan-list (LIST
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Exterior
                                            :task-name 'Build-Exterior
                                            :task-list
                                             '(Lay-Foundation Build-House-Frame))
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Interior-A
                                            :task-name 'Build-Interior
                                            :task-list '(Plumbing))
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Interior-B
                                            :task-name 'Build-Interior
                                            :task-list '(Electricity Decoration))
                                 )
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Lay-Foundation
                                            :agent-class-list
                                             '(John-Class-Lay-Foundation))
                                 (MAKE-TASK :task-name 'Build-House-Frame
                                            :agent-class-list
                                             '(John-Class-Build-House-Frame))
                                 (MAKE-TASK :task-name 'Plumbing
                                            :agent-class-list '(John-Class-Plumbing))
                                 (MAKE-TASK :task-name 'Electricity
                                            :agent-class-list '(John-Class-Electricity))
                                 (MAKE-TASK :task-name 'Decoration
                                            :task-constraints '((CYCLE 24))
                                            :agent-class-list nil)
                                 )
                     :agent-class-list
                                 (LIST
                                  (MAKE-AGENT-CLASS :class-name 'John-Class-Lay-Foundation
                                                    :agent-list
                                                   '(Tom-House-Building Paul-House-Building))
                                  (MAKE-AGENT-CLASS :class-name
                                                    'John-Class-Build-House-Frame
                                                    :agent-list '(Paul-House-Building))
                                  (MAKE-AGENT-CLASS :class-name 'John-Class-Plumbing
                                                    :agent-list '(Tom-House-Building))
                                  (MAKE-AGENT-CLASS :class-name 'John-Class-Electricity
                                                    :agent-list
                                                   '(Tom-House-Building Paul-House-Building))
                                  (MAKE-AGENT-CLASS :class-name 'John-Class-Decoration
                                                    :agent-list
                                                   '(Tom-House-Building Paul-House-Building)))
                     )
      )
)


(defun setup-agent2-d ()
   (setq agent-2 (MAKE-AGENT :agent-name 'John-House-Building
                     :plan-list (LIST
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Exterior
                                            :task-name 'Build-Exterior
                                            :task-list
                                             '(Lay-Foundation Build-House-Frame))
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Interior-A
                                            :task-name 'Build-Interior
                                            :task-list '(Plumbing))
                                 (MAKE-PLAN :plan-name 'Plan-John-Build-Interior-B
                                            :task-name 'Build-Interior
                                            :task-list '(Electricity Decoration))
                                 )
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Lay-Foundation
                                            :agent-class-list
                                             '(John-Class-Lay-Foundation))
                                 (MAKE-TASK :task-name 'Build-House-Frame
                                            :agent-class-list
                                             '(John-Class-Build-House-Frame))
                                 (MAKE-TASK :task-name 'Plumbing
                                            :agent-class-list '(John-Class-Plumbing))
                                 (MAKE-TASK :task-name 'Electricity
                                            :agent-class-list '(John-Class-Electricity))
```

```
                                (MAKE-TASK :task-name 'Decoration
                                           :task-constraints '((CYCLE 24))
                                           :agent-class-list nil)
                                )
                        :agent-class-list
                                (LIST
                                 (MAKE-AGENT-CLASS :class-name 'John-Class-Lay-Foundation
                                                   :agent-list
                                                '(Tom-House-Building Paul-House-Building))
                                 (MAKE-AGENT-CLASS :class-name
                                                'John-Class-Build-House-Frame
                                                   :agent-list '(Paul-House-Building))
                                 (MAKE-AGENT-CLASS :class-name 'John-Class-Plumbing
                                                   :agent-list '(Tom-House-Building))
                                 (MAKE-AGENT-CLASS :class-name 'John-Class-Electricity
                                                   :agent-list '(Paul-House-Building))
                                 (MAKE-AGENT-CLASS :class-name 'John-Class-Decoration
                                                   :agent-list
                                                '(Tom-House-Building Paul-House-Building)))
                        )
        )
  )


  (defun setup-agent3-a ()
     (setq agent-3 (MAKE-AGENT :agent-name 'Tom-House-Building
                        :plan-list (LIST
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Lay-Foundation
                                               :task-name 'Lay-Foundation
                                               :task-list
                                          '(Init-Lay-Foundation Final-Lay-Foundation))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Plumbing
                                               :task-name 'Plumbing
                                               :task-list '(Interior-Plumbing))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Electricity
                                               :task-name 'Electricity
                                               :task-list '(Interior-Electricity))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Decoration
                                               :task-name 'Decoration
                                               :task-list '(Interior-Decoration))
                                    )
                        :task-list (LIST
                                    (MAKE-TASK :task-name 'Init-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Final-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Interior-Plumbing
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Electricity
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Decoration
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    )
                        :agent-class-list
                                    (LIST
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Lay-Foundation
                                                       :agent-list '(Doug-House-Building))
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Build-Interior
                                                       :agent-list '(Rick-House-Building)))
                        )
        )
  )


  (defun setup-agent3-b ()
     (setq agent-3 (MAKE-AGENT :agent-name 'Tom-House-Building
                        :plan-list (LIST
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Lay-Foundation
                                               :task-name 'Lay-Foundation
                                               :task-list
                                          '(Init-Lay-Foundation Final-Lay-Foundation))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Plumbing
                                               :task-name 'Plumbing
                                               :task-list '(Interior-Plumbing))
```

```
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Decoration
                                               :task-name 'Decoration
                                               :task-list '(Interior-Decoration))
                                    )
                          :task-list (LIST
                                    (MAKE-TASK :task-name 'Init-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Final-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Interior-Plumbing
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Electricity
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Decoration
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    )
                          :agent-class-list
                                    (LIST
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Lay-Foundation
                                                       :agent-list '(Doug-House-Building))
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Build-Interior
                                                       :agent-list '(Rick-House-Building)))
                          )
        )
  )


  (defun setup-agent3-c ()
     (setq agent-3 (MAKE-AGENT :agent-name 'Tom-House-Building
                          :plan-list (LIST
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Lay-Foundation
                                               :task-name 'Lay-Foundation
                                               :task-list
                                          '(Init-Lay-Foundation Final-Lay-Foundation))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Electricity
                                               :task-name 'Electricity
                                               :task-list '(Interior-Electricity))
                                    (MAKE-PLAN :plan-name 'Plan-Tom-Decoration
                                               :task-name 'Decoration
                                               :task-list '(Interior-Decoration))
                                    )
                          :task-list (LIST
                                    (MAKE-TASK :task-name 'Init-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Final-Lay-Foundation
                                               :agent-class-list '(Tom-Class-Lay-Foundation))
                                    (MAKE-TASK :task-name 'Interior-Plumbing
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Electricity
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    (MAKE-TASK :task-name 'Interior-Decoration
                                               :agent-class-list '(Tom-Class-Build-Interior))
                                    )
                          :agent-class-list
                                    (LIST
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Lay-Foundation
                                                       :agent-list '(Doug-House-Building))
                                     (MAKE-AGENT-CLASS :class-name 'Tom-Class-Build-Interior
                                                       :agent-list '(Rick-House-Building)))
                          )
        )
  )


  (defun setup-agent4-a ()
     (setq agent-4 (MAKE-AGENT :agent-name 'Paul-House-Building
                          :plan-list (LIST
                                    (MAKE-PLAN :plan-name 'Plan-Paul-Build-House-Frame
                                               :task-name 'Build-House-Frame
                                               :task-list '(Build-Roof Build-Wall))
                                    )
                          :task-list (LIST
                                    (MAKE-TASK :task-name 'Build-Roof
                                               :agent-class-list '(Paul-Class-Build-Roof))
                                    (MAKE-TASK :task-name 'Build-Wall
```

```
                                        :agent-class-list '(Paul-Class-Build-Wall))
                        )
                :agent-class-list
                        (LIST
                         (MAKE-AGENT-CLASS :class-name 'Paul-Class-Build-Roof
                                           :agent-list '(Rick-House-Building))
                         (MAKE-AGENT-CLASS :class-name 'Paul-Class-Build-Wall
                                           :agent-list '(Rick-House-Building)))
                )
        )
 )


 (defun setup-agent4-b ()
    (setq agent-4 (MAKE-AGENT :agent-name 'Paul-House-Building
                        :plan-list (LIST
                                    (MAKE-PLAN :plan-name 'Plan-Paul-Build-House-Frame
                                               :task-name 'Build-House-Frame
                                               :task-list '(Build-Roof Build-Wall))
                                    )
                        :task-list (LIST
                                    (MAKE-TASK :task-name 'Electricity
                                               :task-constraints '((CYCLE 23))
                                               :agent-class-list nil)
                                    (MAKE-TASK :task-name 'Build-Roof
                                               :agent-class-list '(Paul-Class-Build-Roof))
                                    (MAKE-TASK :task-name 'Build-Wall
                                               :agent-class-list '(Paul-Class-Build-Wall))
                                    )
                        :agent-class-list
                                    (LIST
                                     (MAKE-AGENT-CLASS :class-name 'Paul-Class-Build-Roof
                                                       :agent-list '(Rick-House-Building))
                                     (MAKE-AGENT-CLASS :class-name 'Paul-Class-Build-Wall
                                                       :agent-list '(Rick-House-Building)))
                        )
       )
)


(defun setup-agent5-a ()
   (setq agent-5 (MAKE-AGENT :agent-name 'Doug-House-Building
                     :plan-list nil
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Init-Lay-Foundation
                                            :task-constraints '((CYCLE 22))
                                            :agent-class-list nil)
                                 (MAKE-TASK :task-name 'Final-Lay-Foundation
                                            :task-constraints '((CYCLE 22))
                                            :agent-class-list nil)
                                )
                     :agent-class-list nil
                     )
      )
)


(defun setup-agent6-a ()
   (setq agent-6 (MAKE-AGENT :agent-name 'Rick-House-Building
                     :plan-list nil
                     :task-list (LIST
                                 (MAKE-TASK :task-name 'Interior-Plumbing
                                            :task-constraints '((CYCLE 25))
                                            :agent-class-list nil)
                                 (MAKE-TASK :task-name 'Interior-Electricity
                                            :task-constraints '((CYCLE 25))
                                            :agent-class-list nil)
                                 (MAKE-TASK :task-name 'Interior-Decoration
                                            :task-constraints '((CYCLE 25))
                                            :agent-class-list nil)
                                 (MAKE-TASK :task-name 'Build-Roof
                                            :task-constraints '((CYCLE 25))
                                            :agent-class-list nil)
                                 (MAKE-TASK :task-name 'Build-Wall
                                            :task-constraints '((CYCLE 25))
```

```
                                                    :agent-class-list nil)
                                    )
                            :agent-class-list nil
                            )
        )
    )


    (defun setup-system-agent-list ()
       (setq *SYSTEM-AGENT-LIST* '((user user)
                                   (Mark-House-Building agent-1)
                                   (John-House-Building agent-2)
                                   (Tom-House-Building agent-3)
                                   (Paul-House-Building agent-4)
                                   (Doug-House-Building agent-5)
                                   (Rick-House-Building agent-6))
       )
    )


    (defun define-dataset-a ()
       (format t "Dataset A is used.")
       (terpri)
       (setup-user)
       (setup-agent1-a)
       (setup-agent2-a)
       (setup-agent3-a)
       (setup-agent4-a)
       (setup-agent5-a)
       (setup-agent6-a)
       (setup-system-agent-list)
    )


    (defun define-dataset-b-1 ()
       (format t "Dataset B1 is used.")
       (terpri)
       (setup-user)
       (setup-agent1-a)
       (setup-agent2-a)
       (setup-agent3-b)
       (setup-agent4-b)
       (setup-agent5-a)
       (setup-agent6-a)
       (setup-system-agent-list)
    )


    (defun define-dataset-b-2 ()
       (format t "Dataset B2 is used.")
       (terpri)
       (setup-user)
       (setup-agent1-a)
       (setup-agent2-a)
       (setup-agent3-b)
       (setup-agent4-a)
       (setup-agent5-a)
       (setup-agent6-a)
       (setup-system-agent-list)
    )


    (defun define-dataset-c-1 ()
       (format t "Dataset C1 is used.")
       (terpri)
       (setup-user)
       (setup-agent1-a)
       (setup-agent2-b)
       (setup-agent3-b)
       (setup-agent4-b)
       (setup-agent5-a)
       (setup-agent6-a)
       (setup-system-agent-list)
    )
```

```
(defun define-dataset-c-2 ()
   (format t "Dataset C2 is used.")
   (terpri)
   (setup-user)
   (setup-agent1-a)
   (setup-agent2-b)
   (setup-agent3-b)
   (setup-agent4-a)
   (setup-agent5-a)
   (setup-agent6-a)
   (setup-system-agent-list)
)


(defun define-dataset-d-1 ()
   (format t "Dataset D1 is used.")
   (terpri)
   (setup-user)
   (setup-agent1-a)
   (setup-agent2-c)
   (setup-agent3-c)
   (setup-agent4-a)
   (setup-agent5-a)
   (setup-agent6-a)
   (setup-system-agent-list)
)


(defun define-dataset-d-2 ()
   (format t "Dataset D2 is used.")
   (terpri)
   (setup-user)
   (setup-agent1-a)
   (setup-agent2-d)
   (setup-agent3-c)
   (setup-agent4-a)
   (setup-agent5-a)
   (setup-agent6-a)
   (setup-system-agent-list)
)
```

Sample Run with Dataset A:

```
? Dataset A is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G464) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G466) of agent (John-House-Building): Task = Build-Exterior
Processing request (G467) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G472) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G473) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G474) of agent (Tom-House-Building): Task = Plumbing
Processing request (G475) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
```

```
      Set task: Electricity status to AWAITING SUBTASK DISTRIBUTION
      Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
      Set task: Interior-Electricity status to AWAITING DISTRIBUTION
  Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)

  Processing agent: Paul-House-Building ...
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)

  Processing agent: Doug-House-Building ...
  Processing request (G480) of agent (Doug-House-Building): Task = Init-Lay-Foundation
  Processing request (G481) of agent (Doug-House-Building): Task = Final-Lay-Foundation
      Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G491) of agent (Rick-House-Building): Task = Interior-Plumbing
  Processing request (G483) of agent (Rick-House-Building): Task = Build-Roof
  Processing request (G484) of agent (Rick-House-Building): Task = Build-Wall
      Set task: Interior-Plumbing cycle to 1 (Max = 25)
      Set task: Build-Roof cycle to 1 (Max = 25)
      Set task: Build-Wall cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 3 (Max = 24)

  Processing agent: Tom-House-Building ...
  Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Electricity)

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G492) of agent (Rick-House-Building): Task = Interior-Electricity
      Set task: Interior-Plumbing cycle to 2 (Max = 25)
      Set task: Build-Roof cycle to 2 (Max = 25)
      Set task: Build-Wall cycle to 2 (Max = 25)
      Set task: Interior-Electricity cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 4 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

  Processing agent: Rick-House-Building ...
      Set task: Interior-Plumbing cycle to 3 (Max = 25)
      Set task: Build-Roof cycle to 3 (Max = 25)
      Set task: Build-Wall cycle to 3 (Max = 25)
      Set task: Interior-Electricity cycle to 2 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 5 (Max = 24)

  Processing agent: Tom-House-Building ...
```

```
Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 4 (Max = 25)
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
    Set task: Interior-Electricity cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 5 (Max = 25)
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
    Set task: Interior-Electricity cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 6 (Max = 25)
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
    Set task: Interior-Electricity cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 7 (Max = 25)
    Set task: Build-Roof cycle to 7 (Max = 25)
    Set task: Build-Wall cycle to 7 (Max = 25)
    Set task: Interior-Electricity cycle to 6 (Max = 25)
```

```
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 9 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 8 (Max = 25)
    Set task: Build-Roof cycle to 8 (Max = 25)
    Set task: Build-Wall cycle to 8 (Max = 25)
    Set task: Interior-Electricity cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 10 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 9 (Max = 25)
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
    Set task: Interior-Electricity cycle to 8 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 11 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 10 (Max = 25)
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
    Set task: Interior-Electricity cycle to 9 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 12 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 11 (Max = 25)
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
    Set task: Interior-Electricity cycle to 10 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 13 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 12 (Max = 25)
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
    Set task: Interior-Electricity cycle to 11 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 14 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 13 (Max = 25)
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
    Set task: Interior-Electricity cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 15 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 14 (Max = 25)
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
    Set task: Interior-Electricity cycle to 13 (Max = 25)
?
Processing agent: User ...
```

```
Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 16 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 15 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 15 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 15 (Max = 25)
    Set task: Build-Roof cycle to 15 (Max = 25)
    Set task: Build-Wall cycle to 15 (Max = 25)
    Set task: Interior-Electricity cycle to 14 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 17 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 16 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 16 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 16 (Max = 25)
    Set task: Build-Roof cycle to 16 (Max = 25)
    Set task: Build-Wall cycle to 16 (Max = 25)
    Set task: Interior-Electricity cycle to 15 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 18 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 17 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 17 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 17 (Max = 25)
    Set task: Build-Roof cycle to 17 (Max = 25)
    Set task: Build-Wall cycle to 17 (Max = 25)
    Set task: Interior-Electricity cycle to 16 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 19 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 18 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 18 (Max = 22)
```

```
Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 18 (Max = 25)
    Set task: Build-Roof cycle to 18 (Max = 25)
    Set task: Build-Wall cycle to 18 (Max = 25)
    Set task: Interior-Electricity cycle to 17 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 20 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 19 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 19 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 19 (Max = 25)
    Set task: Build-Roof cycle to 19 (Max = 25)
    Set task: Build-Wall cycle to 19 (Max = 25)
    Set task: Interior-Electricity cycle to 18 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 21 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 20 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 20 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 20 (Max = 25)
    Set task: Build-Roof cycle to 20 (Max = 25)
    Set task: Build-Wall cycle to 20 (Max = 25)
    Set task: Interior-Electricity cycle to 19 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 22 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 21 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 21 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 21 (Max = 25)
    Set task: Build-Roof cycle to 21 (Max = 25)
    Set task: Build-Wall cycle to 21 (Max = 25)
    Set task: Interior-Electricity cycle to 20 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
```

```
    Set task: Decoration cycle to 23 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 22 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 22 (Max = 22)

  Processing agent: Rick-House-Building ...
      Set task: Interior-Plumbing cycle to 22 (Max = 25)
      Set task: Build-Roof cycle to 22 (Max = 25)
      Set task: Build-Wall cycle to 22 (Max = 25)
      Set task: Interior-Electricity cycle to 21 (Max = 25)
?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 24 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation status to NO PROBLEM
      Set task: Final-Lay-Foundation status to NO PROBLEM
Notification sent from Doug-House-Building to Tom-House-Building: Init-Lay-Foundation (No-
Problem)
Notification sent from Doug-House-Building to Tom-House-Building: Final-Lay-Foundation
(No-Problem)

  Processing agent: Rick-House-Building ...
      Set task: Interior-Plumbing cycle to 23 (Max = 25)
      Set task: Build-Roof cycle to 23 (Max = 25)
      Set task: Build-Wall cycle to 23 (Max = 25)
      Set task: Interior-Electricity cycle to 22 (Max = 25)
?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration status to NO PROBLEM

  Processing agent: Tom-House-Building ...
      Set task: Init-Lay-Foundation status to NO PROBLEM
      Set task: Final-Lay-Foundation status to NO PROBLEM

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...

  Processing agent: Rick-House-Building ...
      Set task: Interior-Plumbing cycle to 24 (Max = 25)
      Set task: Build-Roof cycle to 24 (Max = 25)
      Set task: Build-Wall cycle to 24 (Max = 25)
      Set task: Interior-Electricity cycle to 23 (Max = 25)
?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...

  Processing agent: Tom-House-Building ...
      Set task: Lay-Foundation status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Lay-Foundation (No-
Problem)

  Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing cycle to 25 (Max = 25)
    Set task: Build-Roof cycle to 25 (Max = 25)
    Set task: Build-Wall cycle to 25 (Max = 25)
    Set task: Interior-Electricity cycle to 24 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM
    Set task: Interior-Electricity cycle to 25 (Max = 25)
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Plumbing (No-
Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Roof (No-Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Wall (No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity status to NO PROBLEM
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Electricity
(No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Plumbing status to NO PROBLEM
    Set task: Interior-Electricity status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Plumbing (No-Problem)

Processing agent: Paul-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Build-House-Frame (No-
Problem)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Plumbing status to NO PROBLEM
    Set task: Build-House-Frame status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Electricity status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Electricity (No-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM
    Set task: Electricity status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Exterior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO PROBLEM
Notification sent from Mark-House-Building to User: House-Building (No-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
```

```
Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
   Incoming Notification:
        John-House-Building:  Build-Exterior  (No-Problem)
        John-House-Building:  Build-Interior  (No-Problem)
   Outgoing Notification:
       User:  House-Building  (No-Problem)
   Activity Blackboard:
     Task: House-Building  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Exterior  Status: No-Problem  Current Agent: John-House-Building
     Task: Build-Interior  Status: No-Problem  Current Agent: John-House-Building
? AGENT: John-House-Building
   Plan:
      Task: Build-Exterior
         Sub-Tasks:
                  Lay-Foundation
                  Build-House-Frame

      Task: Build-Interior
         Sub-Tasks:
                  Plumbing
                  Electricity
                  Decoration


   Task:
      Task:  Lay-Foundation
         Agent Classes:  (John-Class-Lay-Foundation)
      Task:  Build-House-Frame
         Agent Classes:  (John-Class-Build-House-Frame)
      Task:  Plumbing
```

```
         Agent Classes:  (John-Class-Plumbing)
      Task:  Electricity
         Agent Classes:  (John-Class-Electricity)
      Task:  Decoration
         Agent Classes:  Nil

   Agent Class:
      Class Name:  John-Class-Lay-Foundation
                   Agents:  (Tom-House-Building Paul-House-Building)
      Class Name:  John-Class-Build-House-Frame
                   Agents:  (Paul-House-Building)
      Class Name:  John-Class-Plumbing
                   Agents:  (Tom-House-Building Paul-House-Building)
      Class Name:  John-Class-Electricity
                   Agents:  (Tom-House-Building Paul-House-Building)
      Class Name:  John-Class-Decoration
                   Agents:  (Tom-House-Building Paul-House-Building)

   Constraints:

   Incoming Requests:
        Mark-House-Building:  Task  (Build-Exterior)
        Mark-House-Building:  Task  (Build-Interior)
   Outgoing Requests:
        Tom-House-Building:  Task  (Lay-Foundation)
        Paul-House-Building:  Task  (Build-House-Frame)
        Tom-House-Building:  Task  (Plumbing)
        Tom-House-Building:  Task  (Electricity)
   Incoming Notification:
        Tom-House-Building:  Lay-Foundation  (No-Problem)
        Tom-House-Building:  Plumbing  (No-Problem)
        Paul-House-Building:  Build-House-Frame  (No-Problem)
        Tom-House-Building:  Electricity  (No-Problem)
   Outgoing Notification:
        Mark-House-Building:  Build-Exterior  (No-Problem)
        Mark-House-Building:  Build-Interior  (No-Problem)
   Activity Blackboard:
     Task: Build-Exterior  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Interior  Status: No-Problem-Replied  Current Agent: Nil
     Task: Lay-Foundation  Status: No-Problem  Current Agent: Tom-House-Building
     Task: Build-House-Frame  Status: No-Problem  Current Agent: Paul-House-Building
     Task: Plumbing  Status: No-Problem  Current Agent: Tom-House-Building
     Task: Electricity  Status: No-Problem  Current Agent: Tom-House-Building
     Task: Decoration  Status: No-Problem  Current Agent: Tom-House-Building
? AGENT: Tom-House-Building
   Plan:
      Task: Lay-Foundation
         Sub-Tasks:
                  Init-Lay-Foundation
                  Final-Lay-Foundation

      Task: Plumbing
         Sub-Tasks:
                  Interior-Plumbing

      Task: Electricity
         Sub-Tasks:
                  Interior-Electricity

      Task: Decoration
         Sub-Tasks:
                  Interior-Decoration


  Task:
     Task:  Init-Lay-Foundation
        Agent Classes:  (Tom-Class-Lay-Foundation)
     Task:  Final-Lay-Foundation
        Agent Classes:  (Tom-Class-Lay-Foundation)
     Task:  Interior-Plumbing
        Agent Classes:  (Tom-Class-Build-Interior)
     Task:  Interior-Electricity
        Agent Classes:  (Tom-Class-Build-Interior)
     Task:  Interior-Decoration
        Agent Classes:  (Tom-Class-Build-Interior)
```

```
    Agent Class:
       Class Name:  Tom-Class-Lay-Foundation
                    Agents:  (Doug-House-Building)
       Class Name:  Tom-Class-Build-Interior
                    Agents:  (Rick-House-Building)

    Constraints:

    Incoming Requests:
         John-House-Building:  Task  (Lay-Foundation)
         John-House-Building:  Task  (Plumbing)
         John-House-Building:  Task  (Electricity)
    Outgoing Requests:
         Doug-House-Building:  Task  (Init-Lay-Foundation)
         Doug-House-Building:  Task  (Final-Lay-Foundation)
         Rick-House-Building:  Task  (Interior-Plumbing)
         Rick-House-Building:  Task  (Interior-Electricity)
    Incoming Notification:
         Doug-House-Building:  Init-Lay-Foundation  (No-Problem)
         Doug-House-Building:  Final-Lay-Foundation  (No-Problem)
         Rick-House-Building:  Interior-Plumbing  (No-Problem)
         Rick-House-Building:  Interior-Electricity  (No-Problem)
    Outgoing Notification:
         John-House-Building:  Lay-Foundation  (No-Problem)
         John-House-Building:  Plumbing  (No-Problem)
         John-House-Building:  Electricity  (No-Problem)
    Activity Blackboard:
      Task: Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
      Task: Init-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
      Task: Final-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
      Task: Plumbing  Status: No-Problem-Replied  Current Agent: Nil
      Task: Electricity  Status: No-Problem-Replied  Current Agent: Nil
      Task: Interior-Plumbing  Status: No-Problem  Current Agent: Rick-House-Building
      Task: Interior-Electricity  Status: No-Problem  Current Agent: Rick-House-Building
? AGENT: Paul-House-Building
   Plan:
      Task: Build-House-Frame
         Sub-Tasks:
                  Build-Roof
                  Build-Wall


   Task:
      Task:  Build-Roof
         Agent Classes:  (Paul-Class-Build-Roof)
      Task:  Build-Wall
         Agent Classes:  (Paul-Class-Build-Wall)

   Agent Class:
      Class Name:  Paul-Class-Build-Roof
                   Agents:  (Rick-House-Building)
      Class Name:  Paul-Class-Build-Wall
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Build-House-Frame)
   Outgoing Requests:
        Rick-House-Building:  Task  (Build-Roof)
        Rick-House-Building:  Task  (Build-Wall)
   Incoming Notification:
        Rick-House-Building:  Build-Roof  (No-Problem)
        Rick-House-Building:  Build-Wall  (No-Problem)
   Outgoing Notification:
        John-House-Building:  Build-House-Frame  (No-Problem)
   Activity Blackboard:
     Task: Build-House-Frame  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Roof  Status: No-Problem  Current Agent: Rick-House-Building
     Task: Build-Wall  Status: No-Problem  Current Agent: Rick-House-Building
? AGENT: Doug-House-Building
   Plan:

   Task:
```

```
        Task:  Init-Lay-Foundation
           Agent Classes:  Nil
        Task:  Final-Lay-Foundation
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Tom-House-Building:  Task  (Init-Lay-Foundation)
          Tom-House-Building:  Task  (Final-Lay-Foundation)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
          Tom-House-Building:  Init-Lay-Foundation  (No-Problem)
          Tom-House-Building:  Final-Lay-Foundation  (No-Problem)
     Activity Blackboard:
       Task: Init-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
       Task: Final-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
? AGENT: Rick-House-Building
     Plan:

     Task:
        Task:  Interior-Plumbing
           Agent Classes:  Nil
        Task:  Interior-Electricity
           Agent Classes:  Nil
        Task:  Interior-Decoration
           Agent Classes:  Nil
        Task:  Build-Roof
           Agent Classes:  Nil
        Task:  Build-Wall
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Tom-House-Building:  Task  (Interior-Plumbing)
          Paul-House-Building:  Task  (Build-Roof)
          Paul-House-Building:  Task  (Build-Wall)
          Tom-House-Building:  Task  (Interior-Electricity)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
          Tom-House-Building:  Interior-Plumbing  (No-Problem)
          Paul-House-Building:  Build-Roof  (No-Problem)
          Paul-House-Building:  Build-Wall  (No-Problem)
          Tom-House-Building:  Interior-Electricity  (No-Problem)
     Activity Blackboard:
       Task: Interior-Plumbing  Status: No-Problem-Replied  Current Agent: Nil
       Task: Build-Roof  Status: No-Problem-Replied  Current Agent: Nil
       Task: Build-Wall  Status: No-Problem-Replied  Current Agent: Nil
       Task: Interior-Electricity  Status: No-Problem-Replied  Current Agent: Nil
?
```

Sample Run with Dataset B1:

```
? Dataset B1 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G153) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G155) of agent (John-House-Building): Task = Build-Exterior
Processing request (G156) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G161) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G162) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G163) of agent (Tom-House-Building): Task = Plumbing
Processing request (G164) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
```

```
      Set task: Electricity status to NO APPLICABLE PLANS
      Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
  Notification sent from Tom-House-Building to John-House-Building: Electricity (Task-
  Problem)

  Processing agent: Paul-House-Building ...
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)

  Processing agent: Doug-House-Building ...
  Processing request (G169) of agent (Doug-House-Building): Task = Init-Lay-Foundation
  Processing request (G170) of agent (Doug-House-Building): Task = Final-Lay-Foundation
      Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G172) of agent (Rick-House-Building): Task = Build-Roof
  Processing request (G173) of agent (Rick-House-Building): Task = Build-Wall
      Set task: Build-Roof cycle to 1 (Max = 25)
      Set task: Build-Wall cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 3 (Max = 24)
      Set task: Electricity status to TASK PROBLEM
  Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

  Processing agent: Tom-House-Building ...
  Processing request (G164) of agent (Tom-House-Building): Clean-Up = Electricity
  Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G180) of agent (Rick-House-Building): Task = Interior-Plumbing
      Set task: Build-Roof cycle to 2 (Max = 25)
      Set task: Build-Wall cycle to 2 (Max = 25)
      Set task: Interior-Plumbing cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
  Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
      Set task: Decoration cycle to 4 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...
  Processing request (G164) of agent (Paul-House-Building): Task = Electricity
      Set task: Electricity cycle to 1 (Max = 23)

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

  Processing agent: Rick-House-Building ...
      Set task: Build-Roof cycle to 3 (Max = 25)
      Set task: Build-Wall cycle to 3 (Max = 25)
      Set task: Interior-Plumbing cycle to 2 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 5 (Max = 24)
```

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 2 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
    Set task: Interior-Plumbing cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 3 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
    Set task: Interior-Plumbing cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 4 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
    Set task: Interior-Plumbing cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 5 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
```

```
        Set task: Build-Wall cycle to 7 (Max = 25)
        Set task: Interior-Plumbing cycle to 6 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 9 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
        Set task: Electricity cycle to 6 (Max = 23)

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 8 (Max = 25)
        Set task: Build-Wall cycle to 8 (Max = 25)
        Set task: Interior-Plumbing cycle to 7 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 10 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
        Set task: Electricity cycle to 7 (Max = 23)

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 9 (Max = 25)
        Set task: Build-Wall cycle to 9 (Max = 25)
        Set task: Interior-Plumbing cycle to 8 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 11 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
        Set task: Electricity cycle to 8 (Max = 23)

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 10 (Max = 25)
        Set task: Build-Wall cycle to 10 (Max = 25)
        Set task: Interior-Plumbing cycle to 9 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 12 (Max = 24)

    Processing agent: Tom-House-Building ...
```

```
Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 9 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
    Set task: Interior-Plumbing cycle to 10 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 13 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 10 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
    Set task: Interior-Plumbing cycle to 11 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 14 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 11 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
    Set task: Interior-Plumbing cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 15 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 12 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
    Set task: Interior-Plumbing cycle to 13 (Max = 25)
?
```

```
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 16 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 13 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 15 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 15 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 15 (Max = 25)
    Set task: Build-Wall cycle to 15 (Max = 25)
    Set task: Interior-Plumbing cycle to 14 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 17 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 14 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 16 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 16 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 16 (Max = 25)
    Set task: Build-Wall cycle to 16 (Max = 25)
    Set task: Interior-Plumbing cycle to 15 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 18 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 15 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 17 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 17 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 17 (Max = 25)
    Set task: Build-Wall cycle to 17 (Max = 25)
    Set task: Interior-Plumbing cycle to 16 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 19 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 16 (Max = 23)
```

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 18 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 18 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 18 (Max = 25)
    Set task: Build-Wall cycle to 18 (Max = 25)
    Set task: Interior-Plumbing cycle to 17 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 20 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 17 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 19 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 19 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 19 (Max = 25)
    Set task: Build-Wall cycle to 19 (Max = 25)
    Set task: Interior-Plumbing cycle to 18 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 21 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 18 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 20 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 20 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 20 (Max = 25)
    Set task: Build-Wall cycle to 20 (Max = 25)
    Set task: Interior-Plumbing cycle to 19 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 22 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 19 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 21 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 21 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 21 (Max = 25)
    Set task: Build-Wall cycle to 21 (Max = 25)
    Set task: Interior-Plumbing cycle to 20 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 23 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 20 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 22 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 22 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 22 (Max = 25)
    Set task: Build-Wall cycle to 22 (Max = 25)
    Set task: Interior-Plumbing cycle to 21 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 24 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 21 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM
Notification sent from Doug-House-Building to Tom-House-Building: Init-Lay-Foundation (No-
Problem)
Notification sent from Doug-House-Building to Tom-House-Building: Final-Lay-Foundation
(No-Problem)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 23 (Max = 25)
    Set task: Build-Wall cycle to 23 (Max = 25)
    Set task: Interior-Plumbing cycle to 22 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 22 (Max = 23)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 24 (Max = 25)
    Set task: Build-Wall cycle to 24 (Max = 25)
    Set task: Interior-Plumbing cycle to 23 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Lay-Foundation (No-
Problem)
```

```
Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 23 (Max = 23)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 25 (Max = 25)
    Set task: Build-Wall cycle to 25 (Max = 25)
    Set task: Interior-Plumbing cycle to 24 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Electricity (No-
Problem)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM
    Set task: Interior-Plumbing cycle to 25 (Max = 25)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Roof (No-Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Wall (No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Plumbing (No-
Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Build-House-Frame (No-
Problem)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Plumbing status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Plumbing (No-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM
    Set task: Plumbing status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Exterior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO PROBLEM
Notification sent from Mark-House-Building to User: House-Building (No-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


    Task:
       Task:  Build-Exterior
          Agent Classes:  (Mark-Class-Build-Exterior)
       Task:  Build-Interior
          Agent Classes:  (Mark-Class-Build-Interior)

    Agent Class:
       Class Name:  Mark-Class-Build-Exterior
                    Agents:  (John-House-Building)
       Class Name:  Mark-Class-Build-Interior
                    Agents:  (John-House-Building)

    Constraints:

    Incoming Requests:
         User:  Task  (House-Building)
    Outgoing Requests:
         John-House-Building:  Task  (Build-Exterior)
         John-House-Building:  Task  (Build-Interior)
    Incoming Notification:
         John-House-Building:  Build-Exterior  (No-Problem)
         John-House-Building:  Build-Interior  (No-Problem)
    Outgoing Notification:
         User:  House-Building  (No-Problem)
    Activity Blackboard:
      Task: House-Building  Status: No-Problem-Replied  Current Agent: Nil
      Task: Build-Exterior  Status: No-Problem  Current Agent: John-House-Building
      Task: Build-Interior  Status: No-Problem  Current Agent: John-House-Building
? AGENT: John-House-Building
   Plan:
      Task: Build-Exterior
         Sub-Tasks:
                  Lay-Foundation
                  Build-House-Frame

      Task: Build-Interior
         Sub-Tasks:
                  Plumbing
                  Electricity
                  Decoration


   Task:
      Task:  Lay-Foundation
         Agent Classes:  (John-Class-Lay-Foundation)
      Task:  Build-House-Frame
         Agent Classes:  (John-Class-Build-House-Frame)
      Task:  Plumbing
         Agent Classes:  (John-Class-Plumbing)
```

```
        Task:  Electricity
           Agent Classes:  (John-Class-Electricity)
        Task:  Decoration
           Agent Classes:  Nil

     Agent Class:
        Class Name:  John-Class-Lay-Foundation
                     Agents:  (Tom-House-Building Paul-House-Building)
        Class Name:  John-Class-Build-House-Frame
                     Agents:  (Paul-House-Building)
        Class Name:  John-Class-Plumbing
                     Agents:  (Tom-House-Building Paul-House-Building)
        Class Name:  John-Class-Electricity
                     Agents:  (Tom-House-Building Paul-House-Building)
        Class Name:  John-Class-Decoration
                     Agents:  (Tom-House-Building Paul-House-Building)

     Constraints:

     Incoming Requests:
          Mark-House-Building:  Task  (Build-Exterior)
          Mark-House-Building:  Task  (Build-Interior)
     Outgoing Requests:
          Tom-House-Building:  Task  (Lay-Foundation)
          Paul-House-Building:  Task  (Build-House-Frame)
          Tom-House-Building:  Task  (Plumbing)
          Tom-House-Building:  Task  (Electricity)
          Tom-House-Building:  Clean-Up  (Electricity)
          Paul-House-Building:  Task  (Electricity)
     Incoming Notification:
          Tom-House-Building:  Electricity  (Task-Problem)
          Tom-House-Building:  Lay-Foundation  (No-Problem)
          Paul-House-Building:  Electricity  (No-Problem)
          Paul-House-Building:  Build-House-Frame  (No-Problem)
          Tom-House-Building:  Plumbing  (No-Problem)
     Outgoing Notification:
          Mark-House-Building:  Build-Exterior  (No-Problem)
          Mark-House-Building:  Build-Interior  (No-Problem)
     Activity Blackboard:
       Task: Build-Exterior  Status: No-Problem-Replied  Current Agent: Nil
       Task: Build-Interior  Status: No-Problem-Replied  Current Agent: Nil
       Task: Lay-Foundation  Status: No-Problem  Current Agent: Tom-House-Building
       Task: Build-House-Frame  Status: No-Problem  Current Agent: Paul-House-Building
       Task: Plumbing  Status: No-Problem  Current Agent: Tom-House-Building
       Task: Electricity  Status: No-Problem  Current Agent: Paul-House-Building
       Task: Decoration  Status: No-Problem  Current Agent: Nil
? AGENT: Tom-House-Building
    Plan:
       Task: Lay-Foundation
          Sub-Tasks:
                   Init-Lay-Foundation
                   Final-Lay-Foundation

       Task: Plumbing
          Sub-Tasks:
                   Interior-Plumbing

       Task: Decoration
          Sub-Tasks:
                   Interior-Decoration


    Task:
       Task:  Init-Lay-Foundation
          Agent Classes:  (Tom-Class-Lay-Foundation)
       Task:  Final-Lay-Foundation
          Agent Classes:  (Tom-Class-Lay-Foundation)
       Task:  Interior-Plumbing
          Agent Classes:  (Tom-Class-Build-Interior)
       Task:  Interior-Electricity
          Agent Classes:  (Tom-Class-Build-Interior)
       Task:  Interior-Decoration
          Agent Classes:  (Tom-Class-Build-Interior)

    Agent Class:
```

```
        Class Name:  Tom-Class-Lay-Foundation
                     Agents:  (Doug-House-Building)
        Class Name:  Tom-Class-Build-Interior
                     Agents:  (Rick-House-Building)

     Constraints:

     Incoming Requests:
          John-House-Building:  Task  (Lay-Foundation)
          John-House-Building:  Task  (Plumbing)
          John-House-Building:  Task  (Electricity)
          John-House-Building:  Clean-Up  (Electricity)
     Outgoing Requests:
          Doug-House-Building:  Task  (Init-Lay-Foundation)
          Doug-House-Building:  Task  (Final-Lay-Foundation)
          Rick-House-Building:  Task  (Interior-Plumbing)
     Incoming Notification:
          Doug-House-Building:  Init-Lay-Foundation  (No-Problem)
          Doug-House-Building:  Final-Lay-Foundation  (No-Problem)
          Rick-House-Building:  Interior-Plumbing  (No-Problem)
     Outgoing Notification:
          John-House-Building:  Electricity  (Task-Problem)
          John-House-Building:  Lay-Foundation  (No-Problem)
          John-House-Building:  Plumbing  (No-Problem)
     Activity Blackboard:
       Task: Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
       Task: Init-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
       Task: Final-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
       Task: Plumbing  Status: No-Problem-Replied  Current Agent: Nil
       Task: Interior-Plumbing  Status: No-Problem  Current Agent: Rick-House-Building
? AGENT: Paul-House-Building
   Plan:
      Task: Build-House-Frame
         Sub-Tasks:
                  Build-Roof
                  Build-Wall


    Task:
       Task:  Electricity
          Agent Classes:  Nil
       Task:  Build-Roof
          Agent Classes:  (Paul-Class-Build-Roof)
       Task:  Build-Wall
          Agent Classes:  (Paul-Class-Build-Wall)

    Agent Class:
       Class Name:  Paul-Class-Build-Roof
                    Agents:  (Rick-House-Building)
       Class Name:  Paul-Class-Build-Wall
                    Agents:  (Rick-House-Building)

    Constraints:

    Incoming Requests:
         John-House-Building:  Task  (Build-House-Frame)
         John-House-Building:  Task  (Electricity)
    Outgoing Requests:
         Rick-House-Building:  Task  (Build-Roof)
         Rick-House-Building:  Task  (Build-Wall)
    Incoming Notification:
         Rick-House-Building:  Build-Roof  (No-Problem)
         Rick-House-Building:  Build-Wall  (No-Problem)
    Outgoing Notification:
         John-House-Building:  Electricity  (No-Problem)
         John-House-Building:  Build-House-Frame  (No-Problem)
    Activity Blackboard:
      Task: Build-House-Frame  Status: No-Problem-Replied  Current Agent: Nil
      Task: Build-Roof  Status: No-Problem  Current Agent: Rick-House-Building
      Task: Build-Wall  Status: No-Problem  Current Agent: Rick-House-Building
      Task: Electricity  Status: No-Problem-Replied  Current Agent: Nil
? AGENT: Doug-House-Building
   Plan:

   Task:
```

```
        Task:  Init-Lay-Foundation
           Agent Classes:  Nil
        Task:  Final-Lay-Foundation
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Tom-House-Building:  Task  (Init-Lay-Foundation)
          Tom-House-Building:  Task  (Final-Lay-Foundation)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
          Tom-House-Building:  Init-Lay-Foundation  (No-Problem)
          Tom-House-Building:  Final-Lay-Foundation  (No-Problem)
     Activity Blackboard:
       Task: Init-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
       Task: Final-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
? AGENT: Rick-House-Building
     Plan:

     Task:
        Task:  Interior-Plumbing
           Agent Classes:  Nil
        Task:  Interior-Electricity
           Agent Classes:  Nil
        Task:  Interior-Decoration
           Agent Classes:  Nil
        Task:  Build-Roof
           Agent Classes:  Nil
        Task:  Build-Wall
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Paul-House-Building:  Task  (Build-Roof)
          Paul-House-Building:  Task  (Build-Wall)
          Tom-House-Building:  Task  (Interior-Plumbing)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
          Paul-House-Building:  Build-Roof  (No-Problem)
          Paul-House-Building:  Build-Wall  (No-Problem)
          Tom-House-Building:  Interior-Plumbing  (No-Problem)
     Activity Blackboard:
       Task: Build-Roof  Status: No-Problem-Replied  Current Agent: Nil
       Task: Build-Wall  Status: No-Problem-Replied  Current Agent: Nil
       Task: Interior-Plumbing  Status: No-Problem-Replied  Current Agent: Nil
?
```

Sample Run with Dataset B2:

```
? Dataset B2 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G205) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G207) of agent (John-House-Building): Task = Build-Exterior
Processing request (G208) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G213) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G214) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G215) of agent (Tom-House-Building): Task = Plumbing
Processing request (G216) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
```

```
      Set task: Electricity status to NO APPLICABLE PLANS
      Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
  Notification sent from Tom-House-Building to John-House-Building: Electricity (Task-
  Problem)

  Processing agent: Paul-House-Building ...
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)

  Processing agent: Doug-House-Building ...
  Processing request (G221) of agent (Doug-House-Building): Task = Init-Lay-Foundation
  Processing request (G222) of agent (Doug-House-Building): Task = Final-Lay-Foundation
      Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G224) of agent (Rick-House-Building): Task = Build-Roof
  Processing request (G225) of agent (Rick-House-Building): Task = Build-Wall
      Set task: Build-Roof cycle to 1 (Max = 25)
      Set task: Build-Wall cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 3 (Max = 24)
      Set task: Electricity status to TASK PROBLEM
  Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

  Processing agent: Tom-House-Building ...
  Processing request (G216) of agent (Tom-House-Building): Clean-Up = Electricity
  Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G232) of agent (Rick-House-Building): Task = Interior-Plumbing
      Set task: Build-Roof cycle to 2 (Max = 25)
      Set task: Build-Wall cycle to 2 (Max = 25)
      Set task: Interior-Plumbing cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
  Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
      Set task: Decoration cycle to 4 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...
  Processing request (G216) of agent (Paul-House-Building): Task = Electricity
      Set task: Electricity status to NO APPLICABLE PLANS

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

  Processing agent: Rick-House-Building ...
      Set task: Build-Roof cycle to 3 (Max = 25)
      Set task: Build-Wall cycle to 3 (Max = 25)
      Set task: Interior-Plumbing cycle to 2 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 5 (Max = 24)
```

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Notification sent from Paul-House-Building to John-House-Building: Electricity (Task-
Problem)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
    Set task: Interior-Plumbing cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G216) of agent (Paul-House-Building): Clean-Up = Electricity

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
    Set task: Interior-Plumbing cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
    Set task: Interior-Plumbing cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)
```

```
Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
    Set task: Build-Wall cycle to 7 (Max = 25)
    Set task: Interior-Plumbing cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 9 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 8 (Max = 25)
    Set task: Build-Wall cycle to 8 (Max = 25)
    Set task: Interior-Plumbing cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (Task-
Problem)
    Set task: Decoration cycle to 10 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
    Set task: Interior-Plumbing cycle to 8 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G208) of agent (John-House-Building): Clean-Up = Build-Interior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G215) of agent (Tom-House-Building): Clean-Up = Plumbing
Request sent from Tom-House-Building to Rick-House-Building: Clean-Up (Interior-Plumbing)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
Processing request (G232) of agent (Rick-House-Building): Clean-Up = Interior-Plumbing
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
?
Processing agent: User ...
```

```
Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to TASK PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO APPLICABLE PLANS

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Notification sent from Mark-House-Building to User: House-Building (Task-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
?
```

```
Processing agent: User ...
Request sent from User to Mark-House-Building: Clean-Up (House-Building)

Processing agent: Mark-House-Building ...
Processing request (G205) of agent (Mark-House-Building): Clean-Up = House-Building
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Exterior)

Processing agent: John-House-Building ...
Processing request (G207) of agent (John-House-Building): Clean-Up = Build-Exterior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G213) of agent (Tom-House-Building): Clean-Up = Lay-Foundation
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Init-Lay-
Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Final-Lay-
Foundation)

Processing agent: Paul-House-Building ...
Processing request (G214) of agent (Paul-House-Building): Clean-Up = Build-House-Frame
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Roof)
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Wall)

Processing agent: Doug-House-Building ...
Processing request (G221) of agent (Doug-House-Building): Clean-Up = Init-Lay-Foundation
Processing request (G222) of agent (Doug-House-Building): Clean-Up = Final-Lay-Foundation

Processing agent: Rick-House-Building ...
Processing request (G224) of agent (Rick-House-Building): Clean-Up = Build-Roof
Processing request (G225) of agent (Rick-House-Building): Clean-Up = Build-Wall
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
        User:  Clean-Up  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Exterior)
```

```
    Incoming Notification:
         John-House-Building:  Build-Interior  (Task-Problem)
    Outgoing Notification:
         User:  House-Building  (Task-Problem)
    Activity Blackboard:
? AGENT: John-House-Building
    Plan:
       Task: Build-Exterior
          Sub-Tasks:
                   Lay-Foundation
                   Build-House-Frame

       Task: Build-Interior
          Sub-Tasks:
                   Plumbing
                   Electricity
                   Decoration


    Task:
       Task:  Lay-Foundation
          Agent Classes:  (John-Class-Lay-Foundation)
       Task:  Build-House-Frame
          Agent Classes:  (John-Class-Build-House-Frame)
       Task:  Plumbing
          Agent Classes:  (John-Class-Plumbing)
       Task:  Electricity
          Agent Classes:  (John-Class-Electricity)
       Task:  Decoration
          Agent Classes:  Nil

    Agent Class:
       Class Name:  John-Class-Lay-Foundation
                    Agents:  (Tom-House-Building Paul-House-Building)
       Class Name:  John-Class-Build-House-Frame
                    Agents:  (Paul-House-Building)
       Class Name:  John-Class-Plumbing
                    Agents:  (Tom-House-Building Paul-House-Building)
       Class Name:  John-Class-Electricity
                    Agents:  (Tom-House-Building Paul-House-Building)
       Class Name:  John-Class-Decoration
                    Agents:  (Tom-House-Building Paul-House-Building)

    Constraints:

    Incoming Requests:
         Mark-House-Building:  Task  (Build-Exterior)
         Mark-House-Building:  Task  (Build-Interior)
         Mark-House-Building:  Clean-Up  (Build-Interior)
         Mark-House-Building:  Clean-Up  (Build-Exterior)
    Outgoing Requests:
         Tom-House-Building:  Task  (Lay-Foundation)
         Paul-House-Building:  Task  (Build-House-Frame)
         Tom-House-Building:  Task  (Plumbing)
         Tom-House-Building:  Task  (Electricity)
         Tom-House-Building:  Clean-Up  (Electricity)
         Paul-House-Building:  Task  (Electricity)
         Paul-House-Building:  Clean-Up  (Electricity)
         Tom-House-Building:  Clean-Up  (Plumbing)
         Tom-House-Building:  Clean-Up  (Lay-Foundation)
         Paul-House-Building:  Clean-Up  (Build-House-Frame)
    Incoming Notification:
         Tom-House-Building:  Electricity  (Task-Problem)
         Paul-House-Building:  Electricity  (Task-Problem)
    Outgoing Notification:
         Mark-House-Building:  Build-Interior  (Task-Problem)
    Activity Blackboard:
? AGENT: Tom-House-Building
    Plan:
       Task: Lay-Foundation
          Sub-Tasks:
                   Init-Lay-Foundation
                   Final-Lay-Foundation

       Task: Plumbing
```

```
         Sub-Tasks:
                  Interior-Plumbing

      Task: Decoration
         Sub-Tasks:
                  Interior-Decoration


   Task:
      Task:  Init-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Final-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Interior-Plumbing
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Electricity
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Decoration
         Agent Classes:  (Tom-Class-Build-Interior)

   Agent Class:
      Class Name:  Tom-Class-Lay-Foundation
                   Agents:  (Doug-House-Building)
      Class Name:  Tom-Class-Build-Interior
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Lay-Foundation)
        John-House-Building:  Task  (Plumbing)
        John-House-Building:  Task  (Electricity)
        John-House-Building:  Clean-Up  (Electricity)
        John-House-Building:  Clean-Up  (Plumbing)
        John-House-Building:  Clean-Up  (Lay-Foundation)
   Outgoing Requests:
        Doug-House-Building:  Task  (Init-Lay-Foundation)
        Doug-House-Building:  Task  (Final-Lay-Foundation)
        Rick-House-Building:  Task  (Interior-Plumbing)
        Rick-House-Building:  Clean-Up  (Interior-Plumbing)
        Doug-House-Building:  Clean-Up  (Init-Lay-Foundation)
        Doug-House-Building:  Clean-Up  (Final-Lay-Foundation)
   Incoming Notification:
   Outgoing Notification:
        John-House-Building:  Electricity  (Task-Problem)
   Activity Blackboard:
? AGENT: Paul-House-Building
   Plan:
      Task: Build-House-Frame
         Sub-Tasks:
                  Build-Roof
                  Build-Wall


   Task:
      Task:  Build-Roof
         Agent Classes:  (Paul-Class-Build-Roof)
      Task:  Build-Wall
         Agent Classes:  (Paul-Class-Build-Wall)

   Agent Class:
      Class Name:  Paul-Class-Build-Roof
                   Agents:  (Rick-House-Building)
      Class Name:  Paul-Class-Build-Wall
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Build-House-Frame)
        John-House-Building:  Task  (Electricity)
        John-House-Building:  Clean-Up  (Electricity)
        John-House-Building:  Clean-Up  (Build-House-Frame)
   Outgoing Requests:
        Rick-House-Building:  Task  (Build-Roof)
```

```
          Rick-House-Building:  Task  (Build-Wall)
          Rick-House-Building:  Clean-Up  (Build-Roof)
          Rick-House-Building:  Clean-Up  (Build-Wall)
     Incoming Notification:
     Outgoing Notification:
          John-House-Building:  Electricity  (Task-Problem)
     Activity Blackboard:
? AGENT: Doug-House-Building
     Plan:

     Task:
        Task:  Init-Lay-Foundation
           Agent Classes:  Nil
        Task:  Final-Lay-Foundation
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Tom-House-Building:  Task  (Init-Lay-Foundation)
          Tom-House-Building:  Task  (Final-Lay-Foundation)
          Tom-House-Building:  Clean-Up  (Init-Lay-Foundation)
          Tom-House-Building:  Clean-Up  (Final-Lay-Foundation)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
     Activity Blackboard:
? AGENT: Rick-House-Building
     Plan:

     Task:
        Task:  Interior-Plumbing
           Agent Classes:  Nil
        Task:  Interior-Electricity
           Agent Classes:  Nil
        Task:  Interior-Decoration
           Agent Classes:  Nil
        Task:  Build-Roof
           Agent Classes:  Nil
        Task:  Build-Wall
           Agent Classes:  Nil

     Agent Class:

     Constraints:

     Incoming Requests:
          Paul-House-Building:  Task  (Build-Roof)
          Paul-House-Building:  Task  (Build-Wall)
          Tom-House-Building:  Task  (Interior-Plumbing)
          Tom-House-Building:  Clean-Up  (Interior-Plumbing)
          Paul-House-Building:  Clean-Up  (Build-Roof)
          Paul-House-Building:  Clean-Up  (Build-Wall)
     Outgoing Requests:
     Incoming Notification:
     Outgoing Notification:
     Activity Blackboard:
?
```

Sample Run with Dataset C1:

```
? Dataset C1 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G260) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G262) of agent (John-House-Building): Task = Build-Exterior
Processing request (G263) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G268) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G269) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G270) of agent (Tom-House-Building): Task = Plumbing
Processing request (G271) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
```

```
      Set task: Electricity status to NO APPLICABLE PLANS
      Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
  Notification sent from Tom-House-Building to John-House-Building: Electricity (Task-
  Problem)

  Processing agent: Paul-House-Building ...
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
  Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)

  Processing agent: Doug-House-Building ...
  Processing request (G276) of agent (Doug-House-Building): Task = Init-Lay-Foundation
  Processing request (G277) of agent (Doug-House-Building): Task = Final-Lay-Foundation
      Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G279) of agent (Rick-House-Building): Task = Build-Roof
  Processing request (G280) of agent (Rick-House-Building): Task = Build-Wall
      Set task: Build-Roof cycle to 1 (Max = 25)
      Set task: Build-Wall cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 3 (Max = 24)
      Set task: Electricity status to TASK PROBLEM
  Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

  Processing agent: Tom-House-Building ...
  Processing request (G271) of agent (Tom-House-Building): Clean-Up = Electricity
  Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)

  Processing agent: Paul-House-Building ...

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

  Processing agent: Rick-House-Building ...
  Processing request (G287) of agent (Rick-House-Building): Task = Interior-Plumbing
      Set task: Build-Roof cycle to 2 (Max = 25)
      Set task: Build-Wall cycle to 2 (Max = 25)
      Set task: Interior-Plumbing cycle to 1 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
  Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
      Set task: Decoration cycle to 4 (Max = 24)

  Processing agent: Tom-House-Building ...

  Processing agent: Paul-House-Building ...
  Processing request (G271) of agent (Paul-House-Building): Task = Electricity
      Set task: Electricity cycle to 1 (Max = 23)

  Processing agent: Doug-House-Building ...
      Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
      Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

  Processing agent: Rick-House-Building ...
      Set task: Build-Roof cycle to 3 (Max = 25)
      Set task: Build-Wall cycle to 3 (Max = 25)
      Set task: Interior-Plumbing cycle to 2 (Max = 25)
  ?
  Processing agent: User ...

  Processing agent: Mark-House-Building ...

  Processing agent: John-House-Building ...
      Set task: Decoration cycle to 5 (Max = 24)
```

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 2 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
    Set task: Interior-Plumbing cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 3 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
    Set task: Interior-Plumbing cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 4 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
    Set task: Interior-Plumbing cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 5 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
```

```
      Set task: Build-Wall cycle to 7 (Max = 25)
      Set task: Interior-Plumbing cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 9 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 6 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 8 (Max = 25)
    Set task: Build-Wall cycle to 8 (Max = 25)
    Set task: Interior-Plumbing cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 10 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 7 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
    Set task: Interior-Plumbing cycle to 8 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 11 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 8 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
    Set task: Interior-Plumbing cycle to 9 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 12 (Max = 24)

Processing agent: Tom-House-Building ...
```

```
   Processing agent: Paul-House-Building ...
       Set task: Electricity cycle to 9 (Max = 23)

   Processing agent: Doug-House-Building ...
       Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
       Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

   Processing agent: Rick-House-Building ...
       Set task: Build-Roof cycle to 11 (Max = 25)
       Set task: Build-Wall cycle to 11 (Max = 25)
       Set task: Interior-Plumbing cycle to 10 (Max = 25)
   ?
   Processing agent: User ...

   Processing agent: Mark-House-Building ...

   Processing agent: John-House-Building ...
       Set task: Decoration cycle to 13 (Max = 24)

   Processing agent: Tom-House-Building ...

   Processing agent: Paul-House-Building ...
       Set task: Electricity cycle to 10 (Max = 23)

   Processing agent: Doug-House-Building ...
       Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
       Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

   Processing agent: Rick-House-Building ...
       Set task: Build-Roof cycle to 12 (Max = 25)
       Set task: Build-Wall cycle to 12 (Max = 25)
       Set task: Interior-Plumbing cycle to 11 (Max = 25)
   ?
   Processing agent: User ...

   Processing agent: Mark-House-Building ...

   Processing agent: John-House-Building ...
       Set task: Decoration cycle to 14 (Max = 24)

   Processing agent: Tom-House-Building ...

   Processing agent: Paul-House-Building ...
       Set task: Electricity cycle to 11 (Max = 23)

   Processing agent: Doug-House-Building ...
       Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
       Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

   Processing agent: Rick-House-Building ...
       Set task: Build-Roof cycle to 13 (Max = 25)
       Set task: Build-Wall cycle to 13 (Max = 25)
       Set task: Interior-Plumbing cycle to 12 (Max = 25)
   ?
   Processing agent: User ...

   Processing agent: Mark-House-Building ...

   Processing agent: John-House-Building ...
       Set task: Decoration cycle to 15 (Max = 24)

   Processing agent: Tom-House-Building ...

   Processing agent: Paul-House-Building ...
       Set task: Electricity cycle to 12 (Max = 23)

   Processing agent: Doug-House-Building ...
       Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
       Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

   Processing agent: Rick-House-Building ...
       Set task: Build-Roof cycle to 14 (Max = 25)
       Set task: Build-Wall cycle to 14 (Max = 25)
       Set task: Interior-Plumbing cycle to 13 (Max = 25)
   ?
```

```
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 16 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 13 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 15 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 15 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 15 (Max = 25)
    Set task: Build-Wall cycle to 15 (Max = 25)
    Set task: Interior-Plumbing cycle to 14 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 17 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 14 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 16 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 16 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 16 (Max = 25)
    Set task: Build-Wall cycle to 16 (Max = 25)
    Set task: Interior-Plumbing cycle to 15 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 18 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 15 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 17 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 17 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 17 (Max = 25)
    Set task: Build-Wall cycle to 17 (Max = 25)
    Set task: Interior-Plumbing cycle to 16 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 19 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 16 (Max = 23)
```

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 18 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 18 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 18 (Max = 25)
    Set task: Build-Wall cycle to 18 (Max = 25)
    Set task: Interior-Plumbing cycle to 17 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 20 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 17 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 19 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 19 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 19 (Max = 25)
    Set task: Build-Wall cycle to 19 (Max = 25)
    Set task: Interior-Plumbing cycle to 18 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 21 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 18 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 20 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 20 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 20 (Max = 25)
    Set task: Build-Wall cycle to 20 (Max = 25)
    Set task: Interior-Plumbing cycle to 19 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 22 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 19 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 21 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 21 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 21 (Max = 25)
    Set task: Build-Wall cycle to 21 (Max = 25)
    Set task: Interior-Plumbing cycle to 20 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Decoration cycle to 23 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 20 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 22 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 22 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 22 (Max = 25)
    Set task: Build-Wall cycle to 22 (Max = 25)
    Set task: Interior-Plumbing cycle to 21 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 24 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 21 (Max = 23)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM
Notification sent from Doug-House-Building to Tom-House-Building: Init-Lay-Foundation (No-
Problem)
Notification sent from Doug-House-Building to Tom-House-Building: Final-Lay-Foundation
(No-Problem)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 23 (Max = 25)
    Set task: Build-Wall cycle to 23 (Max = 25)
    Set task: Interior-Plumbing cycle to 22 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 22 (Max = 23)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 24 (Max = 25)
    Set task: Build-Wall cycle to 24 (Max = 25)
    Set task: Interior-Plumbing cycle to 23 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Lay-Foundation (No-
Problem)
```

```
Processing agent: Paul-House-Building ...
    Set task: Electricity cycle to 23 (Max = 23)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 25 (Max = 25)
    Set task: Build-Wall cycle to 25 (Max = 25)
    Set task: Interior-Plumbing cycle to 24 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Electricity status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Electricity (No-
Problem)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM
    Set task: Interior-Plumbing cycle to 25 (Max = 25)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Roof (No-Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Wall (No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Plumbing (No-
Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Interior-Plumbing status to NO PROBLEM

Processing agent: Paul-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM
Notification sent from Paul-House-Building to John-House-Building: Build-House-Frame (No-
Problem)

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Build-House-Frame status to NO PROBLEM

Processing agent: Tom-House-Building ...
    Set task: Plumbing status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Plumbing (No-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM
    Set task: Plumbing status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Exterior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Exterior status to NO PROBLEM

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO PROBLEM
Notification sent from Mark-House-Building to User: House-Building (No-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
   Incoming Notification:
        John-House-Building:  Build-Exterior  (No-Problem)
        John-House-Building:  Build-Interior  (No-Problem)
   Outgoing Notification:
        User:  House-Building  (No-Problem)
   Activity Blackboard:
     Task: House-Building  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Exterior  Status: No-Problem  Current Agent: John-House-Building
     Task: Build-Interior  Status: No-Problem  Current Agent: John-House-Building
? AGENT: John-House-Building
   Plan:
      Task: Build-Exterior
         Sub-Tasks:
                  Lay-Foundation
                  Build-House-Frame

      Task: Build-Interior
         Sub-Tasks:
                  Plumbing
                  Electricity
                  Decoration


   Task:
      Task:  Lay-Foundation
         Agent Classes:  (John-Class-Lay-Foundation)
      Task:  Build-House-Frame
         Agent Classes:  (John-Class-Build-House-Frame)
      Task:  Plumbing
         Agent Classes:  (John-Class-Plumbing)
```

```
      Task:  Electricity
         Agent Classes:  (John-Class-Electricity-A John-Class-Electricity-B)
      Task:  Decoration
         Agent Classes:  Nil

   Agent Class:
      Class Name:  John-Class-Lay-Foundation
                   Agents:  (Tom-House-Building Paul-House-Building)
      Class Name:  John-Class-Build-House-Frame
                   Agents:  (Paul-House-Building)
      Class Name:  John-Class-Plumbing
                   Agents:  (Tom-House-Building Paul-House-Building)
      Class Name:  John-Class-Electricity-A
                   Agents:  (Tom-House-Building)
      Class Name:  John-Class-Electricity-B
                   Agents:  (Paul-House-Building)
      Class Name:  John-Class-Decoration
                   Agents:  (Tom-House-Building Paul-House-Building)

   Constraints:

   Incoming Requests:
        Mark-House-Building:  Task  (Build-Exterior)
        Mark-House-Building:  Task  (Build-Interior)
   Outgoing Requests:
        Tom-House-Building:  Task  (Lay-Foundation)
        Paul-House-Building:  Task  (Build-House-Frame)
        Tom-House-Building:  Task  (Plumbing)
        Tom-House-Building:  Task  (Electricity)
        Tom-House-Building:  Clean-Up  (Electricity)
        Paul-House-Building:  Task  (Electricity)
   Incoming Notification:
        Tom-House-Building:  Electricity  (Task-Problem)
        Tom-House-Building:  Lay-Foundation  (No-Problem)
        Paul-House-Building:  Electricity  (No-Problem)
        Paul-House-Building:  Build-House-Frame  (No-Problem)
        Tom-House-Building:  Plumbing  (No-Problem)
   Outgoing Notification:
        Mark-House-Building:  Build-Exterior  (No-Problem)
        Mark-House-Building:  Build-Interior  (No-Problem)
   Activity Blackboard:
     Task: Build-Exterior  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Interior  Status: No-Problem-Replied  Current Agent: Nil
     Task: Lay-Foundation  Status: No-Problem  Current Agent: Tom-House-Building
     Task: Build-House-Frame  Status: No-Problem  Current Agent: Paul-House-Building
     Task: Plumbing  Status: No-Problem  Current Agent: Tom-House-Building
     Task: Electricity  Status: No-Problem  Current Agent: Paul-House-Building
     Task: Decoration  Status: No-Problem  Current Agent: Nil
? AGENT: Tom-House-Building
   Plan:
     Task: Lay-Foundation
        Sub-Tasks:
                 Init-Lay-Foundation
                 Final-Lay-Foundation

     Task: Plumbing
        Sub-Tasks:
                 Interior-Plumbing

     Task: Decoration
        Sub-Tasks:
                 Interior-Decoration


  Task:
     Task:  Init-Lay-Foundation
        Agent Classes:  (Tom-Class-Lay-Foundation)
     Task:  Final-Lay-Foundation
        Agent Classes:  (Tom-Class-Lay-Foundation)
     Task:  Interior-Plumbing
        Agent Classes:  (Tom-Class-Build-Interior)
     Task:  Interior-Electricity
        Agent Classes:  (Tom-Class-Build-Interior)
     Task:  Interior-Decoration
        Agent Classes:  (Tom-Class-Build-Interior)
```

```
      Agent Class:
         Class Name:  Tom-Class-Lay-Foundation
                      Agents:  (Doug-House-Building)
         Class Name:  Tom-Class-Build-Interior
                      Agents:  (Rick-House-Building)

      Constraints:

      Incoming Requests:
           John-House-Building:  Task  (Lay-Foundation)
           John-House-Building:  Task  (Plumbing)
           John-House-Building:  Task  (Electricity)
           John-House-Building:  Clean-Up  (Electricity)
      Outgoing Requests:
           Doug-House-Building:  Task  (Init-Lay-Foundation)
           Doug-House-Building:  Task  (Final-Lay-Foundation)
           Rick-House-Building:  Task  (Interior-Plumbing)
      Incoming Notification:
           Doug-House-Building:  Init-Lay-Foundation  (No-Problem)
           Doug-House-Building:  Final-Lay-Foundation  (No-Problem)
           Rick-House-Building:  Interior-Plumbing  (No-Problem)
      Outgoing Notification:
           John-House-Building:  Electricity  (Task-Problem)
           John-House-Building:  Lay-Foundation  (No-Problem)
           John-House-Building:  Plumbing  (No-Problem)
      Activity Blackboard:
        Task: Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
        Task: Init-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
        Task: Final-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
        Task: Plumbing  Status: No-Problem-Replied  Current Agent: Nil
        Task: Interior-Plumbing  Status: No-Problem  Current Agent: Rick-House-Building
  ? AGENT: Paul-House-Building
     Plan:
        Task: Build-House-Frame
           Sub-Tasks:
                    Build-Roof
                    Build-Wall


      Task:
         Task:  Electricity
            Agent Classes:  Nil
         Task:  Build-Roof
            Agent Classes:  (Paul-Class-Build-Roof)
         Task:  Build-Wall
            Agent Classes:  (Paul-Class-Build-Wall)

      Agent Class:
         Class Name:  Paul-Class-Build-Roof
                      Agents:  (Rick-House-Building)
         Class Name:  Paul-Class-Build-Wall
                      Agents:  (Rick-House-Building)

      Constraints:

      Incoming Requests:
           John-House-Building:  Task  (Build-House-Frame)
           John-House-Building:  Task  (Electricity)
      Outgoing Requests:
           Rick-House-Building:  Task  (Build-Roof)
           Rick-House-Building:  Task  (Build-Wall)
      Incoming Notification:
           Rick-House-Building:  Build-Roof  (No-Problem)
           Rick-House-Building:  Build-Wall  (No-Problem)
      Outgoing Notification:
           John-House-Building:  Electricity  (No-Problem)
           John-House-Building:  Build-House-Frame  (No-Problem)
      Activity Blackboard:
        Task: Build-House-Frame  Status: No-Problem-Replied  Current Agent: Nil
        Task: Build-Roof  Status: No-Problem  Current Agent: Rick-House-Building
        Task: Build-Wall  Status: No-Problem  Current Agent: Rick-House-Building
        Task: Electricity  Status: No-Problem-Replied  Current Agent: Nil
  ? AGENT: Doug-House-Building
     Plan:
```

```
    Task:
       Task:  Init-Lay-Foundation
          Agent Classes:  Nil
       Task:  Final-Lay-Foundation
          Agent Classes:  Nil

    Agent Class:

    Constraints:

    Incoming Requests:
         Tom-House-Building:  Task  (Init-Lay-Foundation)
         Tom-House-Building:  Task  (Final-Lay-Foundation)
    Outgoing Requests:
    Incoming Notification:
    Outgoing Notification:
         Tom-House-Building:  Init-Lay-Foundation  (No-Problem)
         Tom-House-Building:  Final-Lay-Foundation  (No-Problem)
    Activity Blackboard:
      Task: Init-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
      Task: Final-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
? AGENT: Rick-House-Building
   Plan:

   Task:
      Task:  Interior-Plumbing
         Agent Classes:  Nil
      Task:  Interior-Electricity
         Agent Classes:  Nil
      Task:  Interior-Decoration
         Agent Classes:  Nil
      Task:  Build-Roof
         Agent Classes:  Nil
      Task:  Build-Wall
         Agent Classes:  Nil

   Agent Class:

   Constraints:

   Incoming Requests:
        Paul-House-Building:  Task  (Build-Roof)
        Paul-House-Building:  Task  (Build-Wall)
        Tom-House-Building:  Task  (Interior-Plumbing)
   Outgoing Requests:
   Incoming Notification:
   Outgoing Notification:
        Paul-House-Building:  Build-Roof  (No-Problem)
        Paul-House-Building:  Build-Wall  (No-Problem)
        Tom-House-Building:  Interior-Plumbing  (No-Problem)
  Activity Blackboard:
    Task: Build-Roof  Status: No-Problem-Replied  Current Agent: Nil
    Task: Build-Wall  Status: No-Problem-Replied  Current Agent: Nil
    Task: Interior-Plumbing  Status: No-Problem-Replied  Current Agent: Nil
?
```

Sample Run with Dataset C2:

```
? Dataset C2 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G312) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G314) of agent (John-House-Building): Task = Build-Exterior
Processing request (G315) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G320) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G321) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G322) of agent (Tom-House-Building): Task = Plumbing
Processing request (G323) of agent (Tom-House-Building): Task = Electricity
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to AWAITING SUBTASK DISTRIBUTION
```

```
        Set task: Electricity status to NO APPLICABLE PLANS
        Set task: Interior-Plumbing status to AWAITING DISTRIBUTION
    Notification sent from Tom-House-Building to John-House-Building: Electricity (Task-
    Problem)

    Processing agent: Paul-House-Building ...
    Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
    Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)

    Processing agent: Doug-House-Building ...
    Processing request (G328) of agent (Doug-House-Building): Task = Init-Lay-Foundation
    Processing request (G329) of agent (Doug-House-Building): Task = Final-Lay-Foundation
        Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

    Processing agent: Rick-House-Building ...
    Processing request (G331) of agent (Rick-House-Building): Task = Build-Roof
    Processing request (G332) of agent (Rick-House-Building): Task = Build-Wall
        Set task: Build-Roof cycle to 1 (Max = 25)
        Set task: Build-Wall cycle to 1 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 3 (Max = 24)
        Set task: Electricity status to TASK PROBLEM
    Request sent from John-House-Building to Tom-House-Building: Clean-Up (Electricity)

    Processing agent: Tom-House-Building ...
    Processing request (G323) of agent (Tom-House-Building): Clean-Up = Electricity
    Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Plumbing)

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

    Processing agent: Rick-House-Building ...
    Processing request (G339) of agent (Rick-House-Building): Task = Interior-Plumbing
        Set task: Build-Roof cycle to 2 (Max = 25)
        Set task: Build-Wall cycle to 2 (Max = 25)
        Set task: Interior-Plumbing cycle to 1 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
    Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
        Set task: Decoration cycle to 4 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
    Processing request (G323) of agent (Paul-House-Building): Task = Electricity
        Set task: Electricity status to NO APPLICABLE PLANS

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 3 (Max = 25)
        Set task: Build-Wall cycle to 3 (Max = 25)
        Set task: Interior-Plumbing cycle to 2 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 5 (Max = 24)
```

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Notification sent from Paul-House-Building to John-House-Building: Electricity (Task-
Problem)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
    Set task: Interior-Plumbing cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G323) of agent (Paul-House-Building): Clean-Up = Electricity

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
    Set task: Interior-Plumbing cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
    Set task: Interior-Plumbing cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)
```

```
Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
    Set task: Build-Wall cycle to 7 (Max = 25)
    Set task: Interior-Plumbing cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 9 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 8 (Max = 25)
    Set task: Build-Wall cycle to 8 (Max = 25)
    Set task: Interior-Plumbing cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (Task-
Problem)
    Set task: Decoration cycle to 10 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
    Set task: Interior-Plumbing cycle to 8 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G315) of agent (John-House-Building): Clean-Up = Build-Interior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G322) of agent (Tom-House-Building): Clean-Up = Plumbing
Request sent from Tom-House-Building to Rick-House-Building: Clean-Up (Interior-Plumbing)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
Processing request (G339) of agent (Rick-House-Building): Clean-Up = Interior-Plumbing
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
?
Processing agent: User ...
```

```
Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to TASK PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO APPLICABLE PLANS

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Notification sent from Mark-House-Building to User: House-Building (Task-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
?
```

```
Processing agent: User ...
Request sent from User to Mark-House-Building: Clean-Up (House-Building)

Processing agent: Mark-House-Building ...
Processing request (G312) of agent (Mark-House-Building): Clean-Up = House-Building
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Exterior)

Processing agent: John-House-Building ...
Processing request (G314) of agent (John-House-Building): Clean-Up = Build-Exterior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G320) of agent (Tom-House-Building): Clean-Up = Lay-Foundation
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Init-Lay-
Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Final-Lay-
Foundation)

Processing agent: Paul-House-Building ...
Processing request (G321) of agent (Paul-House-Building): Clean-Up = Build-House-Frame
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Roof)
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Wall)

Processing agent: Doug-House-Building ...
Processing request (G328) of agent (Doug-House-Building): Clean-Up = Init-Lay-Foundation
Processing request (G329) of agent (Doug-House-Building): Clean-Up = Final-Lay-Foundation

Processing agent: Rick-House-Building ...
Processing request (G331) of agent (Rick-House-Building): Clean-Up = Build-Roof
Processing request (G332) of agent (Rick-House-Building): Clean-Up = Build-Wall
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
        User:  Clean-Up  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Exterior)
```

```
    Incoming Notification:
         John-House-Building:  Build-Interior  (Task-Problem)
    Outgoing Notification:
         User:  House-Building  (Task-Problem)
    Activity Blackboard:
? AGENT: John-House-Building
    Plan:
       Task: Build-Exterior
          Sub-Tasks:
                   Lay-Foundation
                   Build-House-Frame

       Task: Build-Interior
          Sub-Tasks:
                   Plumbing
                   Electricity
                   Decoration


  Task:
     Task:  Lay-Foundation
        Agent Classes:  (John-Class-Lay-Foundation)
     Task:  Build-House-Frame
        Agent Classes:  (John-Class-Build-House-Frame)
     Task:  Plumbing
        Agent Classes:  (John-Class-Plumbing)
     Task:  Electricity
        Agent Classes:  (John-Class-Electricity-A John-Class-Electricity-B)
     Task:  Decoration
        Agent Classes:  Nil

  Agent Class:
     Class Name:  John-Class-Lay-Foundation
                  Agents:  (Tom-House-Building Paul-House-Building)
     Class Name:  John-Class-Build-House-Frame
                  Agents:  (Paul-House-Building)
     Class Name:  John-Class-Plumbing
                  Agents:  (Tom-House-Building Paul-House-Building)
     Class Name:  John-Class-Electricity-A
                  Agents:  (Tom-House-Building)
     Class Name:  John-Class-Electricity-B
                  Agents:  (Paul-House-Building)
     Class Name:  John-Class-Decoration
                  Agents:  (Tom-House-Building Paul-House-Building)

  Constraints:

  Incoming Requests:
       Mark-House-Building:  Task  (Build-Exterior)
       Mark-House-Building:  Task  (Build-Interior)
       Mark-House-Building:  Clean-Up  (Build-Interior)
       Mark-House-Building:  Clean-Up  (Build-Exterior)
  Outgoing Requests:
       Tom-House-Building:  Task  (Lay-Foundation)
       Paul-House-Building:  Task  (Build-House-Frame)
       Tom-House-Building:  Task  (Plumbing)
       Tom-House-Building:  Task  (Electricity)
       Tom-House-Building:  Clean-Up  (Electricity)
       Paul-House-Building:  Task  (Electricity)
       Paul-House-Building:  Clean-Up  (Electricity)
       Tom-House-Building:  Clean-Up  (Plumbing)
       Tom-House-Building:  Clean-Up  (Lay-Foundation)
       Paul-House-Building:  Clean-Up  (Build-House-Frame)
  Incoming Notification:
       Tom-House-Building:  Electricity  (Task-Problem)
       Paul-House-Building:  Electricity  (Task-Problem)
  Outgoing Notification:
      Mark-House-Building:  Build-Interior  (Task-Problem)
  Activity Blackboard:
? AGENT: Tom-House-Building
  Plan:
     Task: Lay-Foundation
        Sub-Tasks:
                 Init-Lay-Foundation
                 Final-Lay-Foundation
```

```
      Task: Plumbing
         Sub-Tasks:
                  Interior-Plumbing

      Task: Decoration
         Sub-Tasks:
                  Interior-Decoration


   Task:
      Task:  Init-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Final-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Interior-Plumbing
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Electricity
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Decoration
         Agent Classes:  (Tom-Class-Build-Interior)

   Agent Class:
      Class Name:  Tom-Class-Lay-Foundation
                   Agents:  (Doug-House-Building)
      Class Name:  Tom-Class-Build-Interior
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Lay-Foundation)
        John-House-Building:  Task  (Plumbing)
        John-House-Building:  Task  (Electricity)
        John-House-Building:  Clean-Up  (Electricity)
        John-House-Building:  Clean-Up  (Plumbing)
        John-House-Building:  Clean-Up  (Lay-Foundation)
   Outgoing Requests:
        Doug-House-Building:  Task  (Init-Lay-Foundation)
        Doug-House-Building:  Task  (Final-Lay-Foundation)
        Rick-House-Building:  Task  (Interior-Plumbing)
        Rick-House-Building:  Clean-Up  (Interior-Plumbing)
        Doug-House-Building:  Clean-Up  (Init-Lay-Foundation)
        Doug-House-Building:  Clean-Up  (Final-Lay-Foundation)
   Incoming Notification:
   Outgoing Notification:
        John-House-Building:  Electricity  (Task-Problem)
   Activity Blackboard:
? AGENT: Paul-House-Building
   Plan:
      Task: Build-House-Frame
         Sub-Tasks:
                  Build-Roof
                  Build-Wall


   Task:
      Task:  Build-Roof
         Agent Classes:  (Paul-Class-Build-Roof)
      Task:  Build-Wall
         Agent Classes:  (Paul-Class-Build-Wall)

   Agent Class:
      Class Name:  Paul-Class-Build-Roof
                   Agents:  (Rick-House-Building)
      Class Name:  Paul-Class-Build-Wall
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Build-House-Frame)
        John-House-Building:  Task  (Electricity)
        John-House-Building:  Clean-Up  (Electricity)
        John-House-Building:  Clean-Up  (Build-House-Frame)
```

```
    Outgoing Requests:
         Rick-House-Building:  Task  (Build-Roof)
         Rick-House-Building:  Task  (Build-Wall)
         Rick-House-Building:  Clean-Up  (Build-Roof)
         Rick-House-Building:  Clean-Up  (Build-Wall)
    Incoming Notification:
    Outgoing Notification:
         John-House-Building:  Electricity  (Task-Problem)
    Activity Blackboard:
? AGENT: Doug-House-Building
    Plan:

    Task:
       Task:  Init-Lay-Foundation
          Agent Classes:  Nil
       Task:  Final-Lay-Foundation
          Agent Classes:  Nil

    Agent Class:

    Constraints:

    Incoming Requests:
         Tom-House-Building:  Task  (Init-Lay-Foundation)
         Tom-House-Building:  Task  (Final-Lay-Foundation)
         Tom-House-Building:  Clean-Up  (Init-Lay-Foundation)
         Tom-House-Building:  Clean-Up  (Final-Lay-Foundation)
    Outgoing Requests:
    Incoming Notification:
    Outgoing Notification:
    Activity Blackboard:
? AGENT: Rick-House-Building
    Plan:

    Task:
       Task:  Interior-Plumbing
          Agent Classes:  Nil
       Task:  Interior-Electricity
          Agent Classes:  Nil
       Task:  Interior-Decoration
          Agent Classes:  Nil
       Task:  Build-Roof
          Agent Classes:  Nil
       Task:  Build-Wall
          Agent Classes:  Nil

    Agent Class:

    Constraints:

    Incoming Requests:
         Paul-House-Building:  Task  (Build-Roof)
         Paul-House-Building:  Task  (Build-Wall)
         Tom-House-Building:  Task  (Interior-Plumbing)
         Tom-House-Building:  Clean-Up  (Interior-Plumbing)
         Paul-House-Building:  Clean-Up  (Build-Roof)
         Paul-House-Building:  Clean-Up  (Build-Wall)
    Outgoing Requests:
    Incoming Notification:
    Outgoing Notification:
    Activity Blackboard:
?
```

Sample Run with Dataset D1:

```
? Dataset D1 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G367) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G369) of agent (John-House-Building): Task = Build-Exterior
Processing request (G370) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G375) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G376) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G377) of agent (Tom-House-Building): Task = Plumbing
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to NO APPLICABLE PLANS

Processing agent: Paul-House-Building ...
Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)
```

```
Processing agent: Doug-House-Building ...
Processing request (G381) of agent (Doug-House-Building): Task = Init-Lay-Foundation
Processing request (G382) of agent (Doug-House-Building): Task = Final-Lay-Foundation
    Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

Processing agent: Rick-House-Building ...
Processing request (G384) of agent (Rick-House-Building): Task = Build-Roof
Processing request (G385) of agent (Rick-House-Building): Task = Build-Wall
    Set task: Build-Roof cycle to 1 (Max = 25)
    Set task: Build-Wall cycle to 1 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
Notification sent from Tom-House-Building to John-House-Building: Plumbing (Task-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 2 (Max = 25)
    Set task: Build-Wall cycle to 2 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Plumbing status to TASK PROBLEM
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G377) of agent (Tom-House-Building): Clean-Up = Plumbing

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 3 (Max = 25)
    Set task: Build-Wall cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Plumbing status to NO APPLICABLE PLANS

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
    Set task: Build-Wall cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...
Processing request (G398) of agent (Tom-House-Building): Task = Electricity
    Set task: Electricity status to AWAITING SUBTASK DISTRIBUTION
    Set task: Interior-Electricity status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 8 (Max = 25)
```

```
    Set task: Build-Wall cycle to 8 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 3 (Max = 24)

Processing agent: Tom-House-Building ...
Request sent from Tom-House-Building to Rick-House-Building: Task (Interior-Electricity)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
Processing request (G402) of agent (Rick-House-Building): Task = Interior-Electricity
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
    Set task: Interior-Electricity cycle to 1 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 4 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
    Set task: Interior-Electricity cycle to 2 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 5 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
    Set task: Interior-Electricity cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 6 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
    Set task: Interior-Electricity cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
    Set task: Interior-Electricity cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
    Set task: Interior-Electricity cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 9 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 15 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 15 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 15 (Max = 25)
    Set task: Build-Wall cycle to 15 (Max = 25)
    Set task: Interior-Electricity cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 10 (Max = 24)
```

```
    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 16 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 16 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 16 (Max = 25)
        Set task: Build-Wall cycle to 16 (Max = 25)
        Set task: Interior-Electricity cycle to 8 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 11 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 17 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 17 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 17 (Max = 25)
        Set task: Build-Wall cycle to 17 (Max = 25)
        Set task: Interior-Electricity cycle to 9 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 12 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 18 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 18 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 18 (Max = 25)
        Set task: Build-Wall cycle to 18 (Max = 25)
        Set task: Interior-Electricity cycle to 10 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 13 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...
        Set task: Init-Lay-Foundation cycle to 19 (Max = 22)
        Set task: Final-Lay-Foundation cycle to 19 (Max = 22)

    Processing agent: Rick-House-Building ...
        Set task: Build-Roof cycle to 19 (Max = 25)
        Set task: Build-Wall cycle to 19 (Max = 25)
        Set task: Interior-Electricity cycle to 11 (Max = 25)
    ?
    Processing agent: User ...
```

```
Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 14 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 20 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 20 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 20 (Max = 25)
    Set task: Build-Wall cycle to 20 (Max = 25)
    Set task: Interior-Electricity cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 15 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 21 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 21 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 21 (Max = 25)
    Set task: Build-Wall cycle to 21 (Max = 25)
    Set task: Interior-Electricity cycle to 13 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 16 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 22 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 22 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 22 (Max = 25)
    Set task: Build-Wall cycle to 22 (Max = 25)
    Set task: Interior-Electricity cycle to 14 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 17 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM
Notification sent from Doug-House-Building to Tom-House-Building: Init-Lay-Foundation (No-
Problem)
```

```
Notification sent from Doug-House-Building to Tom-House-Building: Final-Lay-Foundation
(No-Problem)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 23 (Max = 25)
    Set task: Build-Wall cycle to 23 (Max = 25)
    Set task: Interior-Electricity cycle to 15 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 18 (Max = 24)

Processing agent: Tom-House-Building ...
    Set task: Init-Lay-Foundation status to NO PROBLEM
    Set task: Final-Lay-Foundation status to NO PROBLEM

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 24 (Max = 25)
    Set task: Build-Wall cycle to 24 (Max = 25)
    Set task: Interior-Electricity cycle to 16 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 19 (Max = 24)

Processing agent: Tom-House-Building ...
    Set task: Lay-Foundation status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Lay-Foundation (No-
Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 25 (Max = 25)
    Set task: Build-Wall cycle to 25 (Max = 25)
    Set task: Interior-Electricity cycle to 17 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 20 (Max = 24)
    Set task: Lay-Foundation status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Build-Roof status to NO PROBLEM
    Set task: Build-Wall status to NO PROBLEM
    Set task: Interior-Electricity cycle to 18 (Max = 25)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Roof (No-Problem)
Notification sent from Rick-House-Building to Paul-House-Building: Build-Wall (No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
```

```
        Set task: Decoration cycle to 21 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
        Set task: Build-Roof status to NO PROBLEM
        Set task: Build-Wall status to NO PROBLEM

    Processing agent: Doug-House-Building ...

    Processing agent: Rick-House-Building ...
        Set task: Interior-Electricity cycle to 19 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 22 (Max = 24)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...
        Set task: Build-House-Frame status to NO PROBLEM
    Notification sent from Paul-House-Building to John-House-Building: Build-House-Frame (No-
    Problem)

    Processing agent: Doug-House-Building ...

    Processing agent: Rick-House-Building ...
        Set task: Interior-Electricity cycle to 20 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Decoration cycle to 23 (Max = 24)
        Set task: Build-House-Frame status to NO PROBLEM

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...

    Processing agent: Rick-House-Building ...
        Set task: Interior-Electricity cycle to 21 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...

    Processing agent: John-House-Building ...
        Set task: Build-Exterior status to NO PROBLEM
        Set task: Decoration cycle to 24 (Max = 24)
    Notification sent from John-House-Building to Mark-House-Building: Build-Exterior (No-
    Problem)

    Processing agent: Tom-House-Building ...

    Processing agent: Paul-House-Building ...

    Processing agent: Doug-House-Building ...

    Processing agent: Rick-House-Building ...
        Set task: Interior-Electricity cycle to 22 (Max = 25)
    ?
    Processing agent: User ...

    Processing agent: Mark-House-Building ...
        Set task: Build-Exterior status to NO PROBLEM

    Processing agent: John-House-Building ...
        Set task: Decoration status to NO PROBLEM
```

```
Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity cycle to 23 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity cycle to 24 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity cycle to 25 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
    Set task: Interior-Electricity status to NO PROBLEM
Notification sent from Rick-House-Building to Tom-House-Building: Interior-Electricity
(No-Problem)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Interior-Electricity status to NO PROBLEM

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
    Set task: Electricity status to NO PROBLEM
Notification sent from Tom-House-Building to John-House-Building: Electricity (No-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO PROBLEM
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (No-
Problem)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO PROBLEM
Notification sent from Mark-House-Building to User: House-Building (No-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
```

```
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
   Incoming Notification:
        John-House-Building:  Build-Exterior  (No-Problem)
        John-House-Building:  Build-Interior  (No-Problem)
   Outgoing Notification:
        User:  House-Building  (No-Problem)
   Activity Blackboard:
     Task: House-Building  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Exterior  Status: No-Problem  Current Agent: John-House-Building
     Task: Build-Interior  Status: No-Problem  Current Agent: John-House-Building
? AGENT: John-House-Building
   Plan:
      Task: Build-Exterior
         Sub-Tasks:
                  Lay-Foundation
                  Build-House-Frame

      Task: Build-Interior
         Sub-Tasks:
                  Plumbing

      Task: Build-Interior
         Sub-Tasks:
                  Electricity
                  Decoration


   Task:
      Task:  Lay-Foundation
         Agent Classes:  (John-Class-Lay-Foundation)
      Task:  Build-House-Frame
         Agent Classes:  (John-Class-Build-House-Frame)
      Task:  Plumbing
         Agent Classes:  (John-Class-Plumbing)
      Task:  Electricity
```

```
          Agent Classes:  (John-Class-Electricity)
       Task:  Decoration
          Agent Classes:  Nil

    Agent Class:
       Class Name:  John-Class-Lay-Foundation
                    Agents:  (Tom-House-Building Paul-House-Building)
       Class Name:  John-Class-Build-House-Frame
                    Agents:  (Paul-House-Building)
       Class Name:  John-Class-Plumbing
                    Agents:  (Tom-House-Building)
       Class Name:  John-Class-Electricity
                    Agents:  (Tom-House-Building Paul-House-Building)
       Class Name:  John-Class-Decoration
                    Agents:  (Tom-House-Building Paul-House-Building)

    Constraints:

    Incoming Requests:
         Mark-House-Building:  Task  (Build-Exterior)
         Mark-House-Building:  Task  (Build-Interior)
    Outgoing Requests:
         Tom-House-Building:  Task  (Lay-Foundation)
         Paul-House-Building:  Task  (Build-House-Frame)
         Tom-House-Building:  Task  (Plumbing)
         Tom-House-Building:  Clean-Up  (Plumbing)
         Tom-House-Building:  Task  (Electricity)
    Incoming Notification:
         Tom-House-Building:  Plumbing  (Task-Problem)
         Tom-House-Building:  Lay-Foundation  (No-Problem)
         Paul-House-Building:  Build-House-Frame  (No-Problem)
         Tom-House-Building:  Electricity  (No-Problem)
    Outgoing Notification:
         Mark-House-Building:  Build-Exterior  (No-Problem)
         Mark-House-Building:  Build-Interior  (No-Problem)
    Activity Blackboard:
      Task: Build-Exterior  Status: No-Problem-Replied  Current Agent: Nil
      Task: Build-Interior  Status: No-Problem-Replied  Current Agent: Nil
      Task: Lay-Foundation  Status: No-Problem  Current Agent: Tom-House-Building
      Task: Build-House-Frame  Status: No-Problem  Current Agent: Paul-House-Building
      Task: Electricity  Status: No-Problem  Current Agent: Tom-House-Building
      Task: Decoration  Status: No-Problem  Current Agent: Nil
? AGENT: Tom-House-Building
   Plan:
      Task: Lay-Foundation
         Sub-Tasks:
                  Init-Lay-Foundation
                  Final-Lay-Foundation

      Task: Electricity
         Sub-Tasks:
                  Interior-Electricity

      Task: Decoration
         Sub-Tasks:
                  Interior-Decoration


  Task:
      Task:  Init-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Final-Lay-Foundation
         Agent Classes:  (Tom-Class-Lay-Foundation)
      Task:  Interior-Plumbing
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Electricity
         Agent Classes:  (Tom-Class-Build-Interior)
      Task:  Interior-Decoration
         Agent Classes:  (Tom-Class-Build-Interior)

  Agent Class:
      Class Name:  Tom-Class-Lay-Foundation
                   Agents:  (Doug-House-Building)
      Class Name:  Tom-Class-Build-Interior
                   Agents:  (Rick-House-Building)
```

```
    Constraints:

    Incoming Requests:
         John-House-Building:  Task  (Lay-Foundation)
         John-House-Building:  Task  (Plumbing)
         John-House-Building:  Clean-Up  (Plumbing)
         John-House-Building:  Task  (Electricity)
    Outgoing Requests:
         Doug-House-Building:  Task  (Init-Lay-Foundation)
         Doug-House-Building:  Task  (Final-Lay-Foundation)
         Rick-House-Building:  Task  (Interior-Electricity)
    Incoming Notification:
         Doug-House-Building:  Init-Lay-Foundation  (No-Problem)
         Doug-House-Building:  Final-Lay-Foundation  (No-Problem)
         Rick-House-Building:  Interior-Electricity  (No-Problem)
    Outgoing Notification:
         John-House-Building:  Plumbing  (Task-Problem)
         John-House-Building:  Lay-Foundation  (No-Problem)
         John-House-Building:  Electricity  (No-Problem)
    Activity Blackboard:
      Task: Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
      Task: Init-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
      Task: Final-Lay-Foundation  Status: No-Problem  Current Agent: Doug-House-Building
      Task: Electricity  Status: No-Problem-Replied  Current Agent: Nil
      Task: Interior-Electricity  Status: No-Problem  Current Agent: Rick-House-Building
? AGENT: Paul-House-Building
   Plan:
      Task: Build-House-Frame
         Sub-Tasks:
                  Build-Roof
                  Build-Wall


   Task:
      Task:  Build-Roof
         Agent Classes:  (Paul-Class-Build-Roof)
      Task:  Build-Wall
         Agent Classes:  (Paul-Class-Build-Wall)

   Agent Class:
      Class Name:  Paul-Class-Build-Roof
                   Agents:  (Rick-House-Building)
      Class Name:  Paul-Class-Build-Wall
                   Agents:  (Rick-House-Building)

   Constraints:

   Incoming Requests:
        John-House-Building:  Task  (Build-House-Frame)
   Outgoing Requests:
        Rick-House-Building:  Task  (Build-Roof)
        Rick-House-Building:  Task  (Build-Wall)
   Incoming Notification:
        Rick-House-Building:  Build-Roof  (No-Problem)
        Rick-House-Building:  Build-Wall  (No-Problem)
   Outgoing Notification:
        John-House-Building:  Build-House-Frame  (No-Problem)
   Activity Blackboard:
     Task: Build-House-Frame  Status: No-Problem-Replied  Current Agent: Nil
     Task: Build-Roof  Status: No-Problem  Current Agent: Rick-House-Building
     Task: Build-Wall  Status: No-Problem  Current Agent: Rick-House-Building
? AGENT: Doug-House-Building
   Plan:

   Task:
      Task:  Init-Lay-Foundation
         Agent Classes:  Nil
      Task:  Final-Lay-Foundation
         Agent Classes:  Nil

   Agent Class:

   Constraints:
```

```
    Incoming Requests:
         Tom-House-Building:  Task  (Init-Lay-Foundation)
         Tom-House-Building:  Task  (Final-Lay-Foundation)
    Outgoing Requests:
    Incoming Notification:
    Outgoing Notification:
         Tom-House-Building:  Init-Lay-Foundation  (No-Problem)
         Tom-House-Building:  Final-Lay-Foundation  (No-Problem)
    Activity Blackboard:
      Task: Init-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
      Task: Final-Lay-Foundation  Status: No-Problem-Replied  Current Agent: Nil
? AGENT: Rick-House-Building
    Plan:

    Task:
       Task:  Interior-Plumbing
          Agent Classes:  Nil
       Task:  Interior-Electricity
          Agent Classes:  Nil
       Task:  Interior-Decoration
          Agent Classes:  Nil
       Task:  Build-Roof
          Agent Classes:  Nil
       Task:  Build-Wall
          Agent Classes:  Nil

    Agent Class:

    Constraints:

    Incoming Requests:
         Paul-House-Building:  Task  (Build-Roof)
         Paul-House-Building:  Task  (Build-Wall)
         Tom-House-Building:  Task  (Interior-Electricity)
    Outgoing Requests:
    Incoming Notification:
    Outgoing Notification:
         Paul-House-Building:  Build-Roof  (No-Problem)
         Paul-House-Building:  Build-Wall  (No-Problem)
         Tom-House-Building:  Interior-Electricity  (No-Problem)
    Activity Blackboard:
      Task: Build-Roof  Status: No-Problem-Replied  Current Agent: Nil
      Task: Build-Wall  Status: No-Problem-Replied  Current Agent: Nil
      Task: Interior-Electricity  Status: No-Problem-Replied  Current Agent: Nil
?
```

Sample Run with Dataset D2:

```
? Dataset D2 is used.
? Request sent from User to Mark-House-Building: Task (House-Building)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Processing request (G416) of agent (Mark-House-Building): Task = House-Building
    Set task: House-Building status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Exterior status to AWAITING DISTRIBUTION
    Set task: Build-Interior status to AWAITING DISTRIBUTION

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Request sent from Mark-House-Building to John-House-Building: Task (Build-Exterior)
Request sent from Mark-House-Building to John-House-Building: Task (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G418) of agent (John-House-Building): Task = Build-Exterior
Processing request (G419) of agent (John-House-Building): Task = Build-Interior
    Set task: Build-Exterior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Interior status to AWAITING SUBTASK DISTRIBUTION
    Set task: Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Build-House-Frame status to AWAITING DISTRIBUTION
    Set task: Plumbing status to AWAITING DISTRIBUTION
Request sent from John-House-Building to Tom-House-Building: Task (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Task (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G424) of agent (Tom-House-Building): Task = Lay-Foundation
    Set task: Lay-Foundation status to AWAITING SUBTASK DISTRIBUTION
    Set task: Init-Lay-Foundation status to AWAITING DISTRIBUTION
    Set task: Final-Lay-Foundation status to AWAITING DISTRIBUTION

Processing agent: Paul-House-Building ...
Processing request (G425) of agent (Paul-House-Building): Task = Build-House-Frame
    Set task: Build-House-Frame status to AWAITING SUBTASK DISTRIBUTION
    Set task: Build-Roof status to AWAITING DISTRIBUTION
    Set task: Build-Wall status to AWAITING DISTRIBUTION

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Tom-House-Building: Task (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G426) of agent (Tom-House-Building): Task = Plumbing
Request sent from Tom-House-Building to Doug-House-Building: Task (Init-Lay-Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Task (Final-Lay-Foundation)
    Set task: Plumbing status to NO APPLICABLE PLANS

Processing agent: Paul-House-Building ...
Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Roof)
Request sent from Paul-House-Building to Rick-House-Building: Task (Build-Wall)
```

```
Processing agent: Doug-House-Building ...
Processing request (G430) of agent (Doug-House-Building): Task = Init-Lay-Foundation
Processing request (G431) of agent (Doug-House-Building): Task = Final-Lay-Foundation
    Set task: Init-Lay-Foundation cycle to 1 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 1 (Max = 22)

Processing agent: Rick-House-Building ...
Processing request (G433) of agent (Rick-House-Building): Task = Build-Roof
Processing request (G434) of agent (Rick-House-Building): Task = Build-Wall
    Set task: Build-Roof cycle to 1 (Max = 25)
    Set task: Build-Wall cycle to 1 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...
Notification sent from Tom-House-Building to John-House-Building: Plumbing (Task-Problem)

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 2 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 2 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 2 (Max = 25)
    Set task: Build-Wall cycle to 2 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Plumbing status to TASK PROBLEM
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Plumbing)

Processing agent: Tom-House-Building ...
Processing request (G426) of agent (Tom-House-Building): Clean-Up = Plumbing

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 3 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 3 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 3 (Max = 25)
    Set task: Build-Wall cycle to 3 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Plumbing status to NO APPLICABLE PLANS

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 4 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 4 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 4 (Max = 25)
    Set task: Build-Wall cycle to 4 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
```

```
Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 5 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 5 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 5 (Max = 25)
    Set task: Build-Wall cycle to 5 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 6 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 6 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 6 (Max = 25)
    Set task: Build-Wall cycle to 6 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to AWAITING DISTRIBUTION
    Set task: Decoration cycle to 1 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 7 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 7 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 7 (Max = 25)
    Set task: Build-Wall cycle to 7 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Request sent from John-House-Building to Paul-House-Building: Task (Electricity)
    Set task: Decoration cycle to 2 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G447) of agent (Paul-House-Building): Task = Electricity
    Set task: Electricity status to NO APPLICABLE PLANS

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 8 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 8 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 8 (Max = 25)
    Set task: Build-Wall cycle to 8 (Max = 25)
```

```
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 3 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Notification sent from Paul-House-Building to John-House-Building: Electricity (Task-
Problem)

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 9 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 9 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 9 (Max = 25)
    Set task: Build-Wall cycle to 9 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Decoration cycle to 4 (Max = 24)
    Set task: Electricity status to TASK PROBLEM
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Electricity)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
Processing request (G447) of agent (Paul-House-Building): Clean-Up = Electricity

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 10 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 10 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 10 (Max = 25)
    Set task: Build-Wall cycle to 10 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Electricity status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 5 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 11 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 11 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 11 (Max = 25)
    Set task: Build-Wall cycle to 11 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
    Set task: Decoration cycle to 6 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...
```

```
Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 12 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 12 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 12 (Max = 25)
    Set task: Build-Wall cycle to 12 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
    Set task: Decoration cycle to 7 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 13 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 13 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 13 (Max = 25)
    Set task: Build-Wall cycle to 13 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...
Notification sent from John-House-Building to Mark-House-Building: Build-Interior (Task-
Problem)
    Set task: Decoration cycle to 8 (Max = 24)

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 14 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 14 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 14 (Max = 25)
    Set task: Build-Wall cycle to 14 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to TASK PROBLEM
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Interior)

Processing agent: John-House-Building ...
Processing request (G419) of agent (John-House-Building): Clean-Up = Build-Interior

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 15 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 15 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 15 (Max = 25)
    Set task: Build-Wall cycle to 15 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: Build-Interior status to NO APPLICABLE PLANS
```

```
Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 16 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 16 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 16 (Max = 25)
    Set task: Build-Wall cycle to 16 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to TASK PROBLEM

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 17 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 17 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 17 (Max = 25)
    Set task: Build-Wall cycle to 17 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
    Set task: House-Building status to NO APPLICABLE PLANS

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 18 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 18 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 18 (Max = 25)
    Set task: Build-Wall cycle to 18 (Max = 25)
?
Processing agent: User ...

Processing agent: Mark-House-Building ...
Notification sent from Mark-House-Building to User: House-Building (Task-Problem)

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...
    Set task: Init-Lay-Foundation cycle to 19 (Max = 22)
    Set task: Final-Lay-Foundation cycle to 19 (Max = 22)

Processing agent: Rick-House-Building ...
    Set task: Build-Roof cycle to 19 (Max = 25)
    Set task: Build-Wall cycle to 19 (Max = 25)
?
Processing agent: User ...
Request sent from User to Mark-House-Building: Clean-Up (House-Building)
```

```
Processing agent: Mark-House-Building ...
Processing request (G416) of agent (Mark-House-Building): Clean-Up = House-Building
Request sent from Mark-House-Building to John-House-Building: Clean-Up (Build-Exterior)

Processing agent: John-House-Building ...
Processing request (G418) of agent (John-House-Building): Clean-Up = Build-Exterior
Request sent from John-House-Building to Tom-House-Building: Clean-Up (Lay-Foundation)
Request sent from John-House-Building to Paul-House-Building: Clean-Up (Build-House-Frame)

Processing agent: Tom-House-Building ...
Processing request (G424) of agent (Tom-House-Building): Clean-Up = Lay-Foundation
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Init-Lay-
Foundation)
Request sent from Tom-House-Building to Doug-House-Building: Clean-Up (Final-Lay-
Foundation)

Processing agent: Paul-House-Building ...
Processing request (G425) of agent (Paul-House-Building): Clean-Up = Build-House-Frame
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Roof)
Request sent from Paul-House-Building to Rick-House-Building: Clean-Up (Build-Wall)

Processing agent: Doug-House-Building ...
Processing request (G430) of agent (Doug-House-Building): Clean-Up = Init-Lay-Foundation
Processing request (G431) of agent (Doug-House-Building): Clean-Up = Final-Lay-Foundation

Processing agent: Rick-House-Building ...
Processing request (G433) of agent (Rick-House-Building): Clean-Up = Build-Roof
Processing request (G434) of agent (Rick-House-Building): Clean-Up = Build-Wall
?
Processing agent: User ...

Processing agent: Mark-House-Building ...

Processing agent: John-House-Building ...

Processing agent: Tom-House-Building ...

Processing agent: Paul-House-Building ...

Processing agent: Doug-House-Building ...

Processing agent: Rick-House-Building ...
? AGENT: Mark-House-Building
   Plan:
      Task: House-Building
         Sub-Tasks:
                  Build-Exterior
                  Build-Interior


   Task:
      Task:  Build-Exterior
         Agent Classes:  (Mark-Class-Build-Exterior)
      Task:  Build-Interior
         Agent Classes:  (Mark-Class-Build-Interior)

   Agent Class:
      Class Name:  Mark-Class-Build-Exterior
                   Agents:  (John-House-Building)
      Class Name:  Mark-Class-Build-Interior
                   Agents:  (John-House-Building)

   Constraints:

   Incoming Requests:
        User:  Task  (House-Building)
        User:  Clean-Up  (House-Building)
   Outgoing Requests:
        John-House-Building:  Task  (Build-Exterior)
        John-House-Building:  Task  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Interior)
        John-House-Building:  Clean-Up  (Build-Exterior)
   Incoming Notification:
        John-House-Building:  Build-Interior  (Task-Problem)
   Outgoing Notification:
```

```
          User:  House-Building  (Task-Problem)
      Activity Blackboard:
   ? AGENT: John-House-Building
      Plan:
         Task: Build-Exterior
            Sub-Tasks:
                     Lay-Foundation
                     Build-House-Frame

         Task: Build-Interior
            Sub-Tasks:
                     Plumbing

         Task: Build-Interior
            Sub-Tasks:
                     Electricity
                     Decoration


     Task:
        Task:  Lay-Foundation
           Agent Classes:  (John-Class-Lay-Foundation)
        Task:  Build-House-Frame
           Agent Classes:  (John-Class-Build-House-Frame)
        Task:  Plumbing
           Agent Classes:  (John-Class-Plumbing)
        Task:  Electricity
           Agent Classes:  (John-Class-Electricity)
        Task:  Decoration
           Agent Classes:  Nil

     Agent Class:
        Class Name:  John-Class-Lay-Foundation
                     Agents:  (Tom-House-Building Paul-House-Building)
        Class Name:  John-Class-Build-House-Frame
                     Agents:  (Paul-House-Building)
        Class Name:  John-Class-Plumbing
                     Agents:  (Tom-House-Building)
        Class Name:  John-Class-Electricity
                     Agents:  (Paul-House-Building)
        Class Name:  John-Class-Decoration
                     Agents:  (Tom-House-Building Paul-House-Building)

     Constraints:

     Incoming Requests:
          Mark-House-Building:  Task  (Build-Exterior)
          Mark-House-Building:  Task  (Build-Interior)
          Mark-House-Building:  Clean-Up  (Build-Interior)
          Mark-House-Building:  Clean-Up  (Build-Exterior)
     Outgoing Requests:
          Tom-House-Building:  Task  (Lay-Foundation)
          Paul-House-Building:  Task  (Build-House-Frame)
          Tom-House-Building:  Task  (Plumbing)
          Tom-House-Building:  Clean-Up  (Plumbing)
          Paul-House-Building:  Task  (Electricity)
          Paul-House-Building:  Clean-Up  (Electricity)
          Tom-House-Building:  Clean-Up  (Lay-Foundation)
          Paul-House-Building:  Clean-Up  (Build-House-Frame)
     Incoming Notification:
          Tom-House-Building:  Plumbing  (Task-Problem)
          Paul-House-Building:  Electricity  (Task-Problem)
     Outgoing Notification:
          Mark-House-Building:  Build-Interior  (Task-Problem)
     Activity Blackboard:
   ? AGENT: Tom-House-Building
     Plan:
        Task: Lay-Foundation
           Sub-Tasks:
                    Init-Lay-Foundation
                    Final-Lay-Foundation

        Task: Electricity
           Sub-Tasks:
                    Interior-Electricity
```

```
        Task: Decoration
           Sub-Tasks:
                    Interior-Decoration


     Task:
        Task:  Init-Lay-Foundation
           Agent Classes:  (Tom-Class-Lay-Foundation)
        Task:  Final-Lay-Foundation
           Agent Classes:  (Tom-Class-Lay-Foundation)
        Task:  Interior-Plumbing
           Agent Classes:  (Tom-Class-Build-Interior)
        Task:  Interior-Electricity
           Agent Classes:  (Tom-Class-Build-Interior)
        Task:  Interior-Decoration
           Agent Classes:  (Tom-Class-Build-Interior)

     Agent Class:
        Class Name:  Tom-Class-Lay-Foundation
                     Agents:  (Doug-House-Building)
        Class Name:  Tom-Class-Build-Interior
                     Agents:  (Rick-House-Building)

     Constraints:

     Incoming Requests:
          John-House-Building:  Task  (Lay-Foundation)
          John-House-Building:  Task  (Plumbing)
          John-House-Building:  Clean-Up  (Plumbing)
          John-House-Building:  Clean-Up  (Lay-Foundation)
     Outgoing Requests:
          Doug-House-Building:  Task  (Init-Lay-Foundation)
          Doug-House-Building:  Task  (Final-Lay-Foundation)
          Doug-House-Building:  Clean-Up  (Init-Lay-Foundation)
          Doug-House-Building:  Clean-Up  (Final-Lay-Foundation)
     Incoming Notification:
     Outgoing Notification:
          John-House-Building:  Plumbing  (Task-Problem)
     Activity Blackboard:
? AGENT: Paul-House-Building
     Plan:
        Task: Build-House-Frame
           Sub-Tasks:
                    Build-Roof
                    Build-Wall


    Task:
       Task:  Build-Roof
          Agent Classes:  (Paul-Class-Build-Roof)
       Task:  Build-Wall
          Agent Classes:  (Paul-Class-Build-Wall)

    Agent Class:
       Class Name:  Paul-Class-Build-Roof
                    Agents:  (Rick-House-Building)
       Class Name:  Paul-Class-Build-Wall
                    Agents:  (Rick-House-Building)

    Constraints:

    Incoming Requests:
         John-House-Building:  Task  (Build-House-Frame)
         John-House-Building:  Task  (Electricity)
         John-House-Building:  Clean-Up  (Electricity)
         John-House-Building:  Clean-Up  (Build-House-Frame)
    Outgoing Requests:
         Rick-House-Building:  Task  (Build-Roof)
         Rick-House-Building:  Task  (Build-Wall)
         Rick-House-Building:  Clean-Up  (Build-Roof)
         Rick-House-Building:  Clean-Up  (Build-Wall)
    Incoming Notification:
    Outgoing Notification:
         John-House-Building:  Electricity  (Task-Problem)
```

```
   Activity Blackboard:
? AGENT: Doug-House-Building
   Plan:

   Task:
      Task:  Init-Lay-Foundation
         Agent Classes:  Nil
      Task:  Final-Lay-Foundation
         Agent Classes:  Nil

   Agent Class:

   Constraints:

   Incoming Requests:
        Tom-House-Building:  Task  (Init-Lay-Foundation)
        Tom-House-Building:  Task  (Final-Lay-Foundation)
        Tom-House-Building:  Clean-Up  (Init-Lay-Foundation)
        Tom-House-Building:  Clean-Up  (Final-Lay-Foundation)
   Outgoing Requests:
   Incoming Notification:
   Outgoing Notification:
   Activity Blackboard:
? AGENT: Rick-House-Building
   Plan:

   Task:
      Task:  Interior-Plumbing
         Agent Classes:  Nil
      Task:  Interior-Electricity
         Agent Classes:  Nil
      Task:  Interior-Decoration
         Agent Classes:  Nil
      Task:  Build-Roof
         Agent Classes:  Nil
      Task:  Build-Wall
         Agent Classes:  Nil

   Agent Class:

   Constraints:

   Incoming Requests:
        Paul-House-Building:  Task  (Build-Roof)
        Paul-House-Building:  Task  (Build-Wall)
        Paul-House-Building:  Clean-Up  (Build-Roof)
        Paul-House-Building:  Clean-Up  (Build-Wall)
   Outgoing Requests:
   Incoming Notification:
   Outgoing Notification:
   Activity Blackboard:
?
```