A Microworld Model for Multiagent Computer-Aided Process Planning

Kejia Wu

M.Eng., Guangdong University of Technology, 2003

B.Eng., Yantai University, 2000

Thesis Submitted in Partial Fulfillment of

The Requirements for the Degree of

Master of Science

In

Mathematical, Computer, and Physical Sciences

(Computer Science)

The University of Northern British Columbia

August 2009

Your file Votre référence
ISBN: 978-0-494-60817-3
Our file Notre référence
ISBN: 978-0-494-60817-3

This Thesis proposes and investigates a novel framework for the study of multiagent solutions for computer-aided process planning (CAPP) in manufacturing systems. The framework is based on a domain-specific microworld model of CAPP, called the CAPP World. The proposed CAPP World is characterized by a product class, a model of a manufacturing cell, and appropriate adaptation and simplification of CAPP modeling concepts from the literature. These abstractions lead to a collection of specific actions that jointly construct a process plan in CAPP World. The analysis shows that the model meets its design objectives: it is simple, representative of properties and difficulties in real-world CAPP, and suitable for formulation and investigation of multiagent solutions for CAPP, as demonstrated through construction of concrete scenarios. The scenarios address topics such as: agent encapsulation, communication, cooperation, and coordination among team members. The Thesis also identifies some topics for future research.

Contents

1 Introduction **1**

2 Background and Related Work **4**

2.1 Planning 4

2.2 Computer-Aided Process Planning 7

2.3 Multiagent Systems 10

2.4 Multiagent Systems for Computer-Aided Process Planning 13

3 A Microworld Approach to CAPP Modeling **17**

3.1 A Rationale for Microworld Approach to CAPP 17

3.2 The Solution Strategy 19

3.3 CAPP Complexity Factors: Analysis and Abstraction 20

4 The CAPP World Model **24**

4.1 The Product Class 24

4.2 The Manufacturing Cell 29

4.3 Models of Process Plan Structures 33

4.4 Actions in CAPP World 39

5 CAPP World as Framework for MAS Studies 45

5.1 Agent Encapsulation 48

5.2 Communication and Coordination 51

5.3 Scenario 1: Basic Process Plan Construction 53

5.4 Scenarios 2a, 2b: Cooperation and Coordination 58

5.4.1 Scenario 2a 58

5.4.2 Scenario 2b 63

5.5 Scenarios 3a, 3b: Process Plan Improvement 66

5.5.1 Scenario 3a 66

5.5.2 Scenario 3b 72

5.6 Scenario 4: Improving Efficiency of Planning 78

5.7 Scenario 5: Varying Team Composition 79

5.8 Scenario 6: Varying Communication Mechanisms 84

6 Analysis and Evaluation 87

7 Conclusions and Future Work **91**

A Notation **94**

A.1 The Pseudocode Language 94

A.2 Activity Diagrams 97

Bibliography **102**

List of Tables

5.1 The map of pseudocode and UML activity diagram (key elements). . . . 47

A.1 The elements of UML activity diagram in CAPP World. 101

List of Figures

4.1 The stock (a) and a part (b) in CAPP World 26

4.2 Two-dimensional representations of stock and part of Figure 4.1 26

4.3 Hole patterns: (a) allowed and (b) disallowed. 27

4.4 Alternative machining feature decompositions 28

4.5 A processing path in CAPP World 29

4.6 A set-up on a pallet with (a) one, (b) two, and (c) four workpieces. . . . 30

4.7 An HMC drilling a hole in a workpiece (a) and, after a table rotation, in another workpiece (b) within the same set-up. 30

4.8 During set-up processing, an HMC exchanges tools from a prepared collection. 31

4.9 A new set-up and a new tool collection are installed simultaneously. . . . 32

5.1 Scenario 1: The Set-up Designer (SD). 55

5.2 Scenario 1: The Machining Operation Designer (MOD). 56

5.3 Scenario 1: Basic Process Plan Construction 57

5.4 Scenario 2a: The Set-up Designer (SD). 60

5.5 Scenario 2a: The Machining Operation Designer (MOD). 61

5.6 The Activity Diagram of Scenario 2a. 62

5.7 Scenario 2b: The Set-up Designer (SD). 65

5.8 Scenario 2b: The Machining Operation Designer (MOD). 66

5.9 Scenario 3a: The Manager-Strategist (MS). 68

5.10 Scenario 3a: The Evaluator (E). 69

5.11 Scenario 3a: The Set-up Designer (SD). 70

5.12 The Activity Diagram of Scenario 3a. 71

5.13 Scenario 3b: The Manager-Strategist (MS). 74

5.14 Scenario 3b: The Evaluator (E). 75

5.15 Scenario 3b: The Machining Operation Designer (MOD). 76

5.16 The Activity Diagram of Scenario 3b. 77

5.17 Scenario 5: The Set-up Designer (SD). 82

5.18 Scenario 5: The Machining Operation Type Designer (MOTD). 83

5.19 Scenario 5: The Machining Operation Method Designer (MOMD). 84

A.1 **foreach** structure. 98

A.2 Priority **foreach** structure. 98

A.3 **when** structure. 99

A.4 **if** structure. 99

Chapter 1

Introduction

An essential part of the manufacturing cycle is *process planning.* It consists in detailed definition of elementary processing steps within a given manufacturing cell, and their order of precedence, that efficiently transform the raw material into finished products satisfying the design specification. In the case of producing metal parts using cutting technologies, process planning typically involves: analysis of product design; deciding on how parts are to be fixed and combined into set-ups for machining; determining the machining operations, including the type, tool, and cutting parameters for each; and optimizing the process plan to meet the management requirements regarding the production time and cost (Halevi, 2003).

A modern approach of computer-aided process planning (CAPP) seeks to automatically implement the activities of process planning with computer technology, and to integrate them with other automated activities in the manufacturing cycle, such as computer-aided design (CAD) and computer-aided manufacturing (CAM). The development of CAPP is presently less mature than either CAD or CAM, due to the empirical nature of knowledge, major complexity challenges, and limitations of the computing approaches employed (Zhang and Xie, 2007).

Over the last decade, CAPP researchers have become increasingly interested in multi-agent systems (MAS) as a software technology that might help overcome the challenges of automating process planning. Agents are autonomous, intelligent entities (such as robots or software components) that can act both proactively and reactively, have social ability, and can compete or cooperate in pursuit of rational goals. In general, MAS has been a very active research area lately, and is gradually becoming closer to a mainstream software technology (Wooldridge, 2009). Still, the best multiagent solutions for CAPP remain to be identified in future research (Shen et al., 2006).

In my studies of MAS for CAPP, I have noted the absence of a simple unified framework in which the properties of MAS solutions for CAPP could be investigated and compared. In this research, I intend to propose a simplified CAPP microworld model that can serve as a vehicle for exploration of multiagent solutions for CAPP. The idea is that common principles to design CAPP system with MAS paradigm are to be investigated via the CAPP microworld model. Starting with the early work of Minsky and his students during 1960's, such microworld methodology has been proved successful in the Artificial Intelligence (AI) realm. The best known microworld model is the Blocks World, first presented by Terry Winograd in his dissertation in 1971, and from then on employed by many researchers (Russell and Norvig, 2003). The experience with Blocks World demonstrates how basic questions underlying complex real-world problems can be successfully abstracted and studied in a simplified, manageable framework. To utilize the advantage of the microworld approach, I create a simplified CAPP world model to capture the main characteristics of the complex process planning, and construct a CAPP microworld that provides a framework for exploration of the problems and principles underlying the MAS approach to CAPP.

In my case, the research started with an analysis of complexity challenges in real-world CAPP, in order to capture their main properties in a simplified abstract model. The CAPP microworld definition includes a limited selection of part designs, set-up designs, operation types, tools, and cutting parameters, which should be easy to integrate into an experimental MAS framework. The justification of the model then focuses on showing that it is:

1. *simple* enough to allow low-cost experimentation with multiagent scenarios that are easy to manage and analyze;

2. *integral* in the sense of including the main aspects of CAPP (set-up design, operation design, choice of tools and cutting parameters, process plan evaluation and optimization) and their mutual interactions, rather than specializing in a specific aspect;

3. *representative* in the sense of capturing the relevant properties and difficulties characteristic of real-world CAPP; and

4. *suitable* for formulation and investigation of MAS design solutions, including agent role specification and assignment, communication techniques, and teamwork scenarios.

The remaining chapters of the Thesis cover background and related work (Chapter 2), a microworld approach to the problem (Chapter 3), a microworld CAPP model formulation (Chapter 4), a series of multiagent scenarios (Chapter 5), analysis and evaluation of the solution (Chapter 6), and some summaries of our work (Chapter 7).

Chapter 2

Background and Related Work

This chapter presents the necessary background in the areas of general planning (Section 2.1), computer-aided process planning (CAPP) (Section 2.2), multiagent systems (MAS) (Section 2.3), and agent-based CAPP (Section 2.4), with emphasis on work that is closely related to my research.

2.1 Planning

Planning is one of the topics in Artificial Intelligence (AI) whose concern is the realization of linear or partially ordered action sequences that lead to a target state.

Planning algorithms search for optimal path in a graph consisting of representations of states and actions. The first significant planning system is STRIPS invented by Fikes and Nilsson (1971). The control structure of STRIPS was derived from the General Problem Solver, which is a state-space searching model developed by Newell and Simon (Russell and Norvig, 2003). The contributions of STRIPS include the representations of states, actions, and means of constructing plans. Its underlying principles and tools still work for recent planning research.

Planning is a complex task. The formal framework for analyzing the complexity of planning was established by Chapman (1985). The focus of his research was domain-independent planning that is nonlinear (in the sense that a plan is a partially ordered set of actions) and conjunctive (in the sense that a plan must make a conjunctive formula true after it is executed). The objective of a nonlinear conjunctive planning algorithm is to find a plan that achieves multiple goals simultaneously. In this context, Chapman consolidated previous work on nonlinear planning into a rigorously defined algorithm called TWEAK, which he proved to be correct and complete (in the sense that it will find a solution if one exists). In particular, TWEAK could handle some classical planning problems that linear planners could not. Chapman was able to prove that the question of whether a given planning problem has a solution is undecidable in general. His intractability theorem states that the problem of deciding whether a given proposition is necessarily true in a non-linear plan (subject to some technical conditions) is NP-hard. A complexity analysis of STRIPS is given in (Bylander, 1994).

Planning research has been divided into two types: general and domain specific. Approaches that seek to understand and solve the planning problems without the use of domain-specific knowledge belong to general planning and approaches that directly use domain heuristics belong to domain specific planning (Hendler et al., 1990). Early work in planning contributed to the understanding of the problem and to the development of general planning techniques. As discussed previously, planning is a hard problem, but its complexity can be reduced by introducing domain-specific constrains to limit search (Chapman, 1985). Ginsberg and Geddis (1991) observe that there are two kinds of meta-level information in declarative systems: (1) knowledge about the base knowledge itself, called *modal* information; and (2) knowledge about what to do with the base knowledge, called *control* information. They claim that while modal information can be domain-dependent, there is no place for domain-dependent control information; any such information is in fact a conflation of domain-independent control rules and domain-dependent modal facts. Yet this conclusion was challenged by Minton (1996) who argued that empirical, domain-dependent control knowledge could play a valuable role. Empirical knowledge can demonstrate two aspects of a domain: facts and criteria indicating a good plan (Kautz and Selman, 1998). With domain specific expertise, a planner can effectively shrink its search space (Ernst et al., 1997). Many researchers acknowledge that domain specific planning promises a more practical direction than general planning (Bacchus and Kabanza, 2000; Penberthy and Weld, 1992; Slaney and Thiebaux, 1996; Weld, 1994).

To investigate complex problems, researchers make the real world easier to study by modeling it. Models are abstract representations of the world that only contain the essential elements relevant to the problem being studied. Highly simplified models used in AI research have become known as *microworlds*. A microworld should be a small but complete subset of reality in which researchers can study a specific domain (Rieber, 1992). The best known microworld is the Blocks World, initially introduced by Terry Winograd in 1971, and later used in simplified form for AI research in several areas, including general planning. The (simplified) Blocks World domain consists of a set of solid cube-shaped blocks (of identical size) on a tabletop, and a robot arm that can pick up a block and place it on top of another block (assuming the top is clear) or on the table. The goal is to build one or more block stacks starting from a given initial state as specified in the planning task. Among other applications, Blocks World is often used to illustrate behaviour of rational agents, e.g. in Wooldridge (2009).

2.2 Computer-Aided Process Planning

Process planning is a manufacturing activity that determines in detail the processing steps of machines in the production cell that transform each workpiece type from its initial to its final state (Halevi, 2003). The final product of this activity is the *process plan* that should achieve given planning requirements, such as lower production cost, or shorter processing time, and satisfy a set of domain constraints (Shen et al., 2006). A good process plan not only decreases production costs, but also ensures product quality. Traditional process planning is time consuming work and heavily depends on experience, which results in low global production efficiency. A computer-aided process planning (CAPP) system uses automated planning techniques to cope with manufacturing process planning. In relation to the entire production cycle, CAPP acts as an interface between two other manufacturing activities: computer-aided design (CAD), automated process of defining the product, and computer-aided manufacturing (CAM), the automated shop floor control. Compared to these two activities, process planning is currently less automated and it is a weak link in the emerging integrated manufacturing.

Planning of machining processes is a complex problem. In the classic approach, its main characteristics come from the domain's strict hierarchical structure of tasks where the final approval decision-making is done on the higher levels and design decision-making is done on the lowest levels (Bose, 1999). Machining processes take place in the physical settings that is characterized by many machine tools, machining operations, cutting tools, and fixtures. The planning of these processes requires a variety of different expert knowledge. According to the expertise required, process planning generally includes the following activities (not necessarily in the order listed): machining feature recognition, machine selection, machining operation selection and design, set-up formation and machining operation sequencing, fixture selection and design, and production cost estimation (Halevi, 2003; Zhao et al., 2000).

A machining feature is a simple-shaped volumetric element to be removed from the stock during manufacturing, along with additional specifications describing the tolerances, surface finish, relationships to other features, etc. Some features must be removed before others (because of geometric accessibility and other reasons) leading to a feature precedence relationship. Feature recognition activity extracts machining features from CAD-generated part designs. Machining features carry information about machining requirements that can facilitate process planning (Mäntylä et al., 1996; Requicha, 1996). For each machining feature, one or more machining operations are selected. Each machining operation is then defined by determining its geometry and selecting a proper cutting tool and cutting parameters. For every machining operation, a machine (or set of machines) capable of efficiently executing it are selected. In order for a machining operation to be executed on the part by a machine, the machining feature that corresponds to that machining operation has to be positioned so as to be accessible for machining. The set-up formation is the activity that determines the position of workpiece in order to make features on the workpiece accessible for machining. Sequencing of machining operations determines the order in which the operations are executed, in compliance with the precedence relationships among the machining features. Fixture selection and design activity determines the fixture tool that keeps a part instance in position for machining, possibly in combination with other part instances within a set-up; it depends on the geometric and material characteristics of the parts as well as on machining features, machining operations, and machines (Scallan, 2003).

Traditionally, there are two major approaches to automated process planning: variant and generative. The variant approach is based on group technology where parts are classified into family groups such that each group has a standard process plan; this approach has been used in industries with few product families and large numbers of parts per family. In the generative approach, the process plan is synthesized from the knowledge about the geometry of the part, its material, production environment, and required production cost. It is suitable for large companies with frequently changing product families of small number of products per family.

Both Bose (1999) and Zhao et al. (2000) give comprehensive surveys of CAPP systems based on the variant and generative approaches using different software system development methods, such as object-oriented and AI based expert systems. Naming the advantages and disadvantages of methods used in applications of both variant and generative CAPP systems in regards to constructing the process plan they conclude that these systems follow the hierarchical nature of planning the machining process, resulting in centralized system architectures that lack flexibility for changes. Due to the complexity of CAPP, a system with such structure can hardly fulfill the role of an efficient integrator of manufacturing processes. They see a solution to these problems in the generative approach, which is implemented by a system architecture that consists of cooperating subsystems. These subsystems exercise greater autonomy in decision-making within their domains of expertise. They also note that the introduction of nonlinear planning methods represents a milestone in process planning, making it possible to decompose a planning problem into sub-problems that can be distributed to different problem solvers. The software solution is seen in using intelligent software agents to implement different sub-problem solvers.

Wang et al. (2006) give an overview of CAPP systems, including CAPP systems based on intelligent agents, and conclude that although the two main objectives of CAPP systems are the generation of an efficient manufacturing process plan and integration of planning activity into a fully automatic manufacturing cycle, most process planning systems are off-line, centralized, and not integrated with related activities such as scheduling. In this Thesis, scheduling is discussed as a supplementary aspect that is helpful to investigate agent-based CAPP modeling, since it has close relationship with process planning in manufacturing.

2.3 Multiagent Systems

There is currently no clear consensus about the precise definition of multiagent system or even individual agent. In their recent monograph on foundations of multiagent systems, Shoham and Leyton-Brown (2009) avoid rigorous definition and state that "multiagent systems are those systems that include multiple autonomous entities with either diverging information or diverging interests or both". Wooldridge and Jennings (1995) define an agent as follows:

> An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

The same authors suggest the following further properties as characteristic of intelligent agents, cited here from Wooldridge (2009):

1. *Reactivity.* Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
2. *Proactiveness.* Intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives.
3. *Social ability.* Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

An agent receives sensory inputs, called percepts, from the environment, and acts on the environment. In addition, an agent can have an internal state that is transformed based on perception and influences the agent's actions. In order to know what to do, an agent needs to know how desirable each state of the environment is. This is typically specified as a utility function that assigns real-number values to states, allowing the agent to assess how useful a particular action would be. Agents can be given tasks, formulated as predicates, asking them to achieve or maintain certain properties of the environment.

Of particular interest from the application point of view is the class of practical reasoning agents, endowed with a state of mind and reasoning mechanisms enabling them to determine what to do and how to achieve it. The main conceptual framework for developing such agents is the *belief-desire-intention (BDI)* framework, based on philosophical ideas of Bratman (1987). An agent acquires beliefs about the environment, uses them to formulate desires, and conducts a deliberation process in which some desires become intentions. Desires are potential objectives, not backed by any commitment, that need not be achievable or mutually consistent. Intentions are the adopted objectives that the agent believes are achievable and mutually consistent. The agent remains committed to them with a degree of persistence, but not irrevocably. The remaining problem of how to achieve the intentions is known as means-ends reasoning or planning. One of the best known BDI systems is *PRS* designed by Georgeff and Lansky (1987). Some later successful intelligent agent systems that employed the same principle are revisions or extensions of PRS. Some of them have already served in the real world for many years, like the OASIS air traffic management system. Starting with the work of Rao and Georgeff, BDI reasoning has been formalized in modal logic, giving rise to a substantial direction of research (Wooldridge, 2009). In practice, BDI reasoning is to some extent implemented with a *plan* concept, which was first presented by Pollack (1992). Such a *plan* is also a critical component of some MAS platforms, such as JADEX (Pokahr et al., 2005).

In multiagent systems there is a wide variety of agent interactions, involving competition, cooperation, and combinations of both. In order to communicate, agents need a common ontology of the discourse domain and a common language. For our purposes, the most important aspect of multiagent systems is the ability of agents to work together as a team towards a common goal, in particular the teamwork of BDI agents, coordinated through joint intentions. A blackboard structure is used to accomplish cooperating joint intentions. The blackboard model is appropriate for cooperative solving problems through diverse knowledge sources in a concurrent and parallel circumstance where sub-solutions contributed by distributed problems solvers usually have the spacial or time uncertainty attribute. Reddy and O'Hare (1991) summarized the following facilities available from the blackboard model:

1. experimental testing of opportunistic strategies;
2. handling of uncertain or continuously varying data;
3. intelligent planning and scheduling of tasks;
4. separation of control knowledge into distinct agents;
5. global consistency checking;
6. hierarchical representation of data.

One of advantages of the blackboard model is suitable for solving ill-defined complex problems, e.g. process planning (Corkill, 1991). The effectiveness of blackboard as a cooperative problems solving mechanism have been reviewed by Corkill (1991); Nii (1986a,b); Reddy and O'Hare (1991). In our CAPP microworld, the blackboard mechanism is adopted to coordinate joint intentions, including design feature sets, machining operation types, and so on, for process planning.

2.4 Multiagent Systems for Computer-Aided Process Planning

Compared to manufacturing activities such as computer-aided design (CAD) or shop-floor control (computer-aided manufacturing—CAM), process planning is currently less automated; it is a weak link in the emerging integrated manufacturing. In their recent survey of distributed process planning, Wang et al. (2006) note that "most process planning systems are off-line, centralized, and not integrated with related activities such as scheduling". In attempts to overcome this situation, a number of research groups have recently built experimental CAPP systems based on multiagent software technology, taking advantage of the newest results of the research progress in multiagent systems. In this section, we review those efforts starting with two recently published surveys of MAS for CAPP research.

Shen et al. (2006) give a comprehensive survey of CAPP systems implemented using agent technology. All of those systems are characterized by fixed multiagent architectures where each agent exhibits well defined expert knowledge for performing a particular activity within the planning process. As major advantages of multiagent based CAPP system they name "modularity, reconfigurability, scalability, upgradeability, and robustness (including fault recovery)". In considering the challenges, their analysis of existing agent-based CAPP systems focuses on four issues related to the realization of these systems.

The first issue is the encapsulation of domain tasks through decomposition in order to determine the system architecture. In the reviewed systems there are two approaches taken in decomposing manufacturing task to determine the system architecture: functional and physical. In the functional approach, the system is decomposed into functional modules that are encapsulated in agents. These modules realize one of the major functions within manufacturing system, such as process planning or scheduling. In the physical approach, the roles of real actors in the manufacturing process, people or machines, are assigned to the intelligent agents.

The second issue is agent modeling. The discussion covers different techniques used in implementing individual agents' capabilities, such as decision-making or learning mechanisms. For team-level decision-making, most of the multiagent systems use coordination or negotiation mechanisms, while individual agents for their local decision-making use knowledge-based mechanisms. For learning, a variety of learning mechanisms have been used such as case-based or neural networks.

The third issue is system structure. The discussion covers different agent systems architectures as organized frameworks within which agents are designed and implemented. They have classified them into three groups: hierarchical, federated, and autonomous. The hierarchical structure is mostly used in agent-based systems that have functional decomposition. In order to avoid the problems inherent to centralized hierarchical systems, most agent-based planning systems use federated architectures, where agents' activities are coordinated via facilitation or mediation, to reduce communication overhead.

The fourth group of issues are coordination and negotiation mechanisms. Systems based on functional decomposition usually need predefined coordination mechanisms. The majority of systems based on physical decomposition use standard negotiation and coordination mechanisms such as Contract Net Protocol or its modifications.

Zhang and Xie (2007) have categorized the MAS applications in process planning into three groups according to the approach the agents take in solving the planning problem: cooperative, blackboard architecture, and integrated approach. In the cooperative approach, the planning problem is solved through cooperation and negotiation of the expert agents using a specialized communication language; in the blackboard approach, the agents with unique expertise cooperate by exchanging the necessary information through a blackboard; and in the integrated approach, the process planning problem is considered as integral part of the complete manufacturing cycle, that includes design and scheduling. The authors conclude that agent technology is suitable for application of collaborative process planning, but it is still a new area that needs substantial further research in order to be used for development of a proper agent-based CAPP system.

Our first observation from these reviews is that the surveyed experimental multiagent CAPP systems exhibit a variety of MAS architectures and design solutions, and yet the surveys provide very little comparison as to which architectures and solutions might be more appropriate than others for application in the CAPP domain, and what are their specific advantages and disadvantages. Indeed, it would appear that such studies would be difficult to conduct, given the fact that these are largely independently conceived and built systems, each addressing its own specific objectives and employing its own fixed set of multiagent solutions. Our second observation is that the reported systems, while serving as vehicles for the advancement of CAPP research, have not as yet resulted in mature industrially deployed systems. Addressing the possible reasons for this, Shen et al. (2006) state in their conclusions: "However, whether the potential advantages of agent-based approaches can actually be realized in industrial systems will depend on the selection of a suitable system architecture for agent organization and an appropriate approach for agent encapsulation; on the design and implementation of effective mechanisms and protocols for communication, cooperation, coordination, and negotiation; and on the design and implementation of advanced internal architectures and efficient decision schemes of individual agents." We conclude that, for those outcomes to occur, one needs to carefully consider the reasons for the current absence of comparative evaluation studies of multiagent solutions for CAPP, and explore what needs to be done to make such studies possible.

Chapter 3

A Microworld Approach to CAPP Modeling

This chapter discusses a microworld approach to computer-aided process planning (CAPP) modeling and how problems in the domain can be investigated by our microworld—the CAPP World. A rationale analysis is presented in Section 3.1, the strategy we employ for modeling CAPP with a microworld approach is discussed in Section 3.2, and the complexity factors we target in the CAPP microworld are investigated in Section 3.3.

3.1 A Rationale for Microworld Approach to CAPP

Our review of recent research on multiagent systems (MAS) for computer-aided process planning (CAPP) in Section 2.4 has led to several observations. First, there has been a significant research progress in the field. A number of experimental systems have been built and investigated. The standardization that provides a unified ontological basis for integrated manufacturing and CAPP in particular has also advanced significantly through efforts such as the STEP project. Second, those developments have not yet resulted in mature industrially deployed systems. According to the authors of a recent comprehensive survey of the field (Shen et al., 2006), the realization of the potential advantages of MAS in practical industrial CAPP systems depends on finding the most suitable multiagent solutions for CAPP. Their conclusions suggest that this includes: appropriate agent encapsulation; the design and implementation of advanced internal architectures and efficient decision schemes of individual agents; suitable system architectures for agent organization; and the design and implementation of effective mechanisms and protocols for communication, cooperation, coordination, and negotiation. Third, while the designs of concrete experimental systems address many of the multi-agent issues in the above list, there are no major comparative studies regarding the suitability and performance of specific multiagent solutions that they employ. This is understandable, as presently there exists no unified, manageable framework for conducting such studies.

In order to create a framework for analyzable study of multiagent solutions for CAPP, we employ the microworld model approach. The success of microworld model approach has been proved by extensive Artificial Intelligence (AI) research cited in Section 2.1. The underlying reason for its success is that the real world is full of distracting and obscuring details. The most important advantage of employing a microworld model approach is to strip inessential factors and keep the crucial elements of the study problem. In this research, I examine the essential complexity factors in CAPP and propose abstractions that help overcome or manage those complexities for the purposes of multi-agent studies. The proposed abstractions are integrated into a domain-specific microworld—the CAPP World.

3.2 The Solution Strategy

The key questions in addressing the stated problem of the microworld computer-aided process planning (CAPP) model design include the following:

1. How does one recognize the key aspects of CAPP that need to be represented in the model?
2. How does one build a formal model that includes all the necessary elements and yet remains simple enough to allow thorough analysis?
3. How can one show that the defined microworld model is indeed suitable for relevant comparisons of multiagent solutions for CAPP?

In order to keep this difficult problem within the limits of an M.Sc. thesis, I intend to focus on demonstrating the feasibility and viability of the concept of microworld for CAPP, rather than trying to devise an ideal all-encompassing microworld model. I intend to define a concrete, simple model that includes a selection of CAPP functions, and show its relevance to multiagent studies. It is understood that this basic model needs to be enhanced or altered to either represent some additional aspects of CAPP or handle some additional aspects of multiagent systems (MAS) design and organization that are beyond the scope of this Thesis.

All specific elements of the microworld model are chosen from the CAPP domain of metal-cutting technologies with chip removal. Those elements involve design features, machine tools, machining features, process operations, set-ups, and so forth. More discussion about this issue is presented in Chapter 4.

3.3 CAPP Complexity Factors: Analysis and Abstraction

As a domain-specific microworld model, CAPP World needs to incorporate suitable abstractions of domain-specific concepts that are relevant to its intended purpose, namely the comparative study of multiagent solutions. In order to decide what aspects of computer-aided process planning (CAPP) need to be captured in the model, I intend to first analyze the factors that make real-world process planning complex and difficult to automate. Polajnar et al. (2008b) identify the following seven components of CAPP complexity:

1. *Combinatorial complexity.* The construction of process plans involves extensive steps with a number of opportunities in every step for each of the multitude of machining operations required by products. Also, set-up formation may be available in a variety of ways. All this results in an explosion of the decision space.

2. *Technological complexity.* The construction, evaluation, and optimization of process plans involve various types of specialist expertise and software support. For example, feature recognition, selection of cutting tools and machining parameters, formation of set-ups, and specification of optimization strategies all depend on different types of skills and experience.

3. *Logical complexity.* The planning decisions are highly interdependent. For example, a choice of cutting parameters may disturb the set-up stability; each intermediate decision influences global performance and may affect the balance of conflicting objectives. This may result in backtracking.

4. *Social complexity.* In modern manufacturing, segments of the production cycle

are increasingly distributed across different organizations in diverse environments. Social interactions among those segments increase CAPP complexity.

5. *Empirical nature of knowledge.* Manufacturing technologies heavily depend on empirical research, with the knowledge base growing rapidly. Organizing, formalizing, and utilizing empirical manufacturing knowledge are challenges for CAPP systems.

6. *Reasoning is hard to formalize.* It is hard for automated systems like CAPP to model human general understanding of technical areas and to capture the intuitive, qualitative, or approximate reasoning skills so as to shrink the decision space substantially.

7. *Decisions with local scope have global effects.* Designers constructing a process plan cannot directly see the global impact of their decisions. Solutions of design sub-problems have narrow scopes. However, there are always global requirements, such as the desired cutting time, the largest utilization of equipment, and so forth, which are affected by intermediate local decisions.

The review for complex aspects in real-world CAPP is rather complete and representative: every important requirement for a comprehensive and robust CAPP system hits in one of the complexity issues. With respect to each of these types of complexity, our analysis focuses on deciding what should be reflected in the model and what should be avoided. With respect to the complexity summarized by Polajnar et al. (2008b), the problem domain of the research is constrained:

1. The model will reduce the combinatorial complexity by defining a relatively small space of relevant design choices, in order to ensure that our simplified analysis of multiagent solutions for CAPP remains tractable and manageable. For instance,

the stock shape of each test case part type will be a cube, at most two tool access directions (TAD) will be allowed, and / or amount of design features will be controlled under some level.

2. CAPP World does not focus on some specific manufacturing expertise and data interaction with some real-world software systems. The ontology of process planning, such as denotation of design and machining features, will be constrained to a limited domain; the construction, evaluation, and optimization of process plans will be formalized in the scaled done environement. For example, some evaluation criteria may involve reducing the number of set-up instances and / or improving the tool usage economy.

3. The model will recognize that the logical complexity of real-world CAPP is a key concern in selecting and evaluating multiagent solutions, and will retain representative inter-dependencies between activities involving different types of technological expertise (e.g., machining operation design vs. set-up formation). This goal can be achieved by defining limited design and machining feature datums.

4. Unless a CAPP World instance is reconfigured to investigate social complexity, this issue will not be considered in our microworld research model. This case involves investigating integrating manufacturing systems which is not the focus of the Thesis. CAPP World will be designed flexible enough to allow reconfiguration for it.

5. To attack the empirical nature of knowledge, we formulate limited actual experience within the agents' knowledge bases, e.g. ontologizing a limited scope of manufacturing knowledge, and formatting empirical messages with agent communication languages (ACL), instead of focusing on organizing and formalizing process planning expertise.

6. As a study vehicle in area of artificial intelligence (AI), it is inevitable to involve reasoning, although this issue for automated systems is hard. We mainly employ belief-desire-intention (BDI) theory as the reasoning engine of the CAPP World, considering that BDI has been proved as an effective methodology in multiagent reasoning research.

7. That decision with local scope have global effects, i.e. decision myopia, is a common disadvantage of distributed approaches (Monostori et al., 2006). This drawback can be compensated by plan-caching and backtracking mechanisms.

Chapter 4

The CAPP World Model

A formalization of the computer-aided process planning (CAPP) microworld, CAPP World, is proposed in this chapter. Section 4.1 specifies a class of products that are suitable for CAPP World; Section 4.2 presents the shop-floor setting, namely a group of machines comprising the manufacturing cell, for CAPP World; the formulation of process plans and underlying models, i.e. part machining, set-up, and process plan, are proposed in Section 4.3; and meta-actions employed in those models mentioned above are listed in Section 4.4.

4.1 The Product Class

The first step in defining the CAPP World model is to specify the class of products that can be manufactured. This section introduces a very restricted product class that still permits modeling of many computer-aided process planning (CAPP) concepts that are relevant to multiagent studies. The section also introduces elements of a formal framework for CAPP adapted from Polajnar et al. (2008a) and Polajnar and Polajnar

(2009).

During the manufacturing process, a block of raw material, called the *stock*, is transformed into a finished *part*. Parts are manufactured in series of identical products. A *series* A_i consists of a part type P_i, and the number of n_i of instances to be produced. A *part type* P_i specifies the stock and the part design, typically generated by a computer-aided design (CAD) system. An instance of the part can then be specified as (P_i, ℓ) where $\ell \in \{1, \dots, n_i\}$ is the instance index within the series. A batch $A = (A_1, \dots, A_n)$ of $n \geq 1$ series to be manufactured together is called an *assortment*.

Having received the part design from a CAD system, a CAPP system must first ensure that the part is represented in terms of machining features. A *machining feature* is a volumetric element to be removed from the stock during manufacturing (such as a hole, pocket, slot, step, etc.) along with additional specifications describing the tolerances, surface finish, relationships to other features, etc. [1]Some features must be removed before others (because of geometric accessibility and other reasons); the design thus specifies a partial ordering on the set of features, called the *feature precedence*. We formally define part type as $P_i = (S_i, F_i, \leq_i)$, where S_i is the stock, F_i the set features, and $\leq_i$ the precedence relation on F_i.

We now define the universe of simple part types in CAPP World. The stock is always a cube of a standard size, with a distinguished central plane (Figure 4.1a). The part can have holes in its sides, as illustrated in Figure 4.1b. Each hole is composed of one or more coaxial cylindrical holes, with diameters decreasing from top to bottom cylinder; its axis is orthogonal to the cube side and lies in the central plane. The bottom hole can be either blind or a through hole. Each cylinder is fully contained within the cube.

[1]The machining feature model of a part type can be generated either by the CAD system or by a front module of the CAPP system. In the sequel it is assumed that the part type design includes the set of machining features, except where stated otherwise.

The holes must not intersect. The fact that the axes of all holes are in the same central

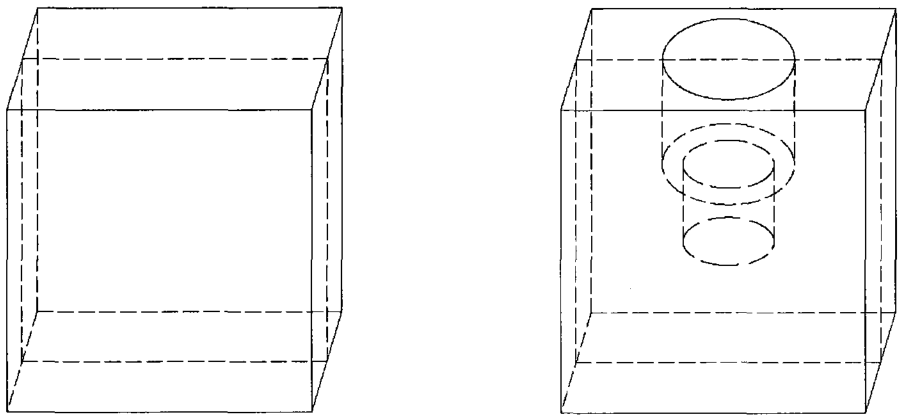

(a) Stock. (b) Part.

Figure 4.1: The stock (a) and a part (b) in CAPP World

plane allows the part design to be represented in two dimensions, simply by showing its central plane cross section, as in Figure 4.2a and 4.2b.

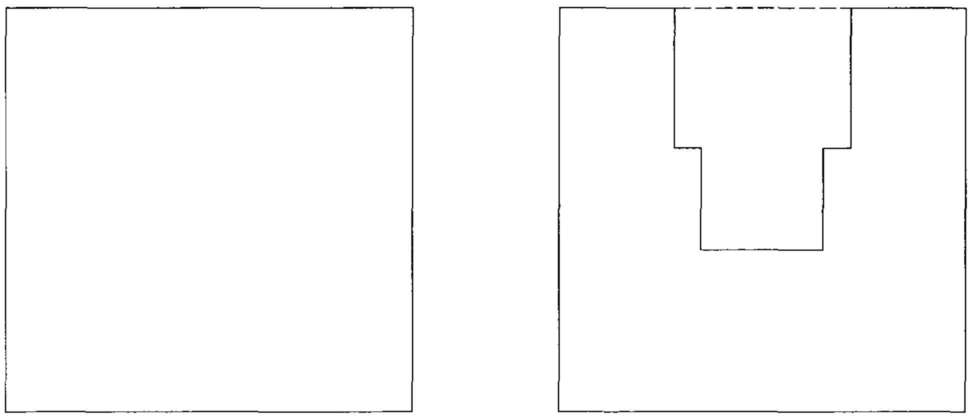
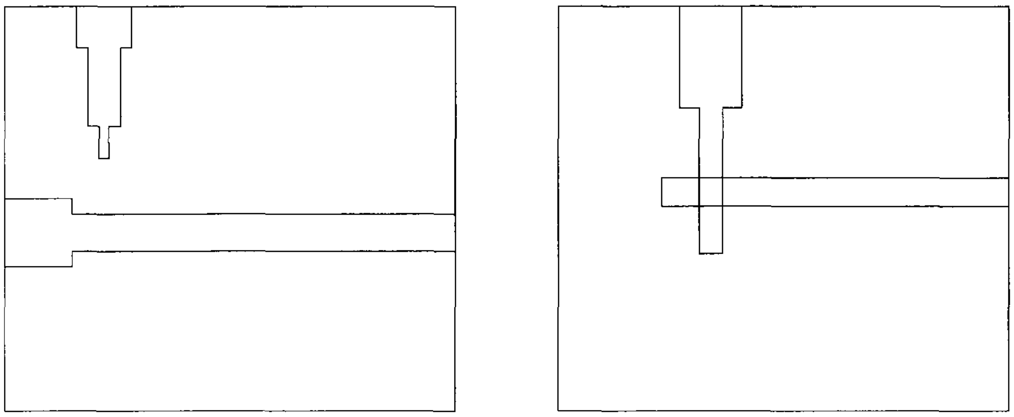

(a) Stock. (b) Part.

Figure 4.2: Two-dimensional representations of stock and part of Figure 4.1

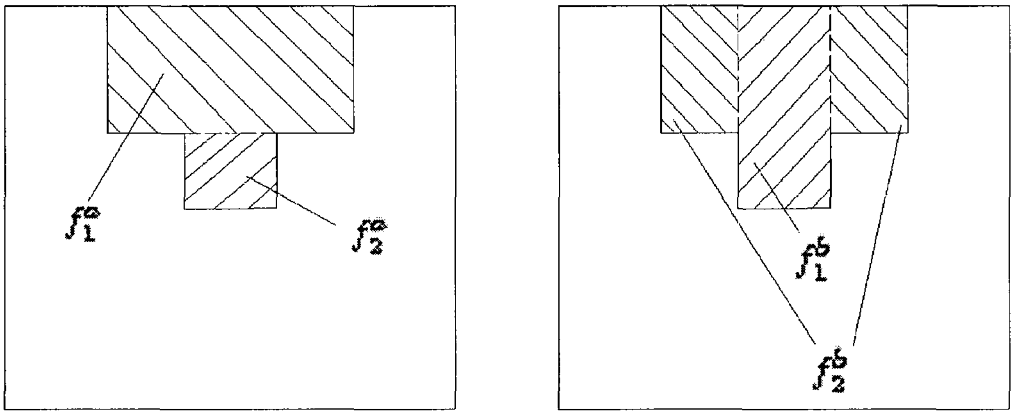
Since all holes have axes in the central plane, the two sides parallel to the central plane do not have any holes. Figure 4.3a and Figure 4.3b illustrate legal and illegal hole patterns.



(a) Allowed hole patterns. (b) Disallowed hole patterns.

Figure 4.3: Hole patterns: (a) allowed and (b) disallowed.

A machining feature can be viewed as a "negative body" to be removed during the machining process. For instance, a cylindrical hole is created by a removal of a solid cylinder. Even though all part designs in CAPP World can be represented in terms of cylindrical holes, from the machining perspective it is not necessarily the best choice to design all features as solid cylinders. Figure 4.4 shows two alternative feature decompositions of the same part type. In Figure 4.4a, features f_1^a and f_2^a are both solid cylinders, while in Figure 4.4b feature f_1^b is a solid cylinder and f_2^b is a hollow cylinder. In general, all machining features in CAPP World are either solid or hollow cylinders. We use the term *workpiece* to denote the states of a part during machining. The initial state corresponds to the stock, and the final state to the finished part. A workpiece type is represented as (P_i, φ), where P_i is a part type and φ is the set of its features that remain to be removed. Thus the stock workpiece type is represented as (P_i, F_i) and the finished part workpiece type as $(P_i, \emptyset)$. To ensure that the state (P_i, φ) is reachable, φ must have the property that is defined next. For a part type P_i with feature set F_i and precedence relationship $\leq_i$, a subset $\varphi \subseteq F_i$ is said to be *precedence-consistent* if, for all $f, g \in F_i$ such that $f \leq_i g$, whenever f belongs to φ so does g. Intuitively, if f must be removed before g, and f is still to be removed, then so is g. The set of all *workpiece*



(a) Decomposition into two solid cylinders. (b) Decomposition into solid and hollow cylinders.

Figure 4.4: Alternative machining feature decompositions

types of P_i is then defined as

$$\mathcal{W}_i = \{(P_i, \varphi) \mid \varphi \subseteq F_i \text{ and } \varphi \text{ is precedence-consistent } \}$$

A part instance (P_i, ℓ) and workpiece type (P_i, φ) determine a *workpiece instance* (P_i, φ, ℓ).

A transition from a workpiece type (P_i, φ_1) to a workpiece type (P_i, φ_2) is called a *processing step* if $\varphi_2 = \varphi_1 \cup \{f\}$ where $f \notin \varphi_1$, i.e., if the transition can be accomplished by a removal of a single feature. A sequence of connected processing steps is called a *processing path* (Figure 4.5).

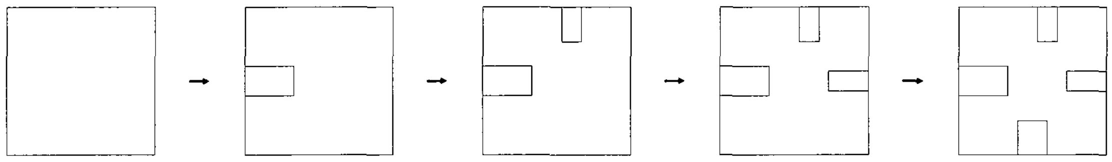


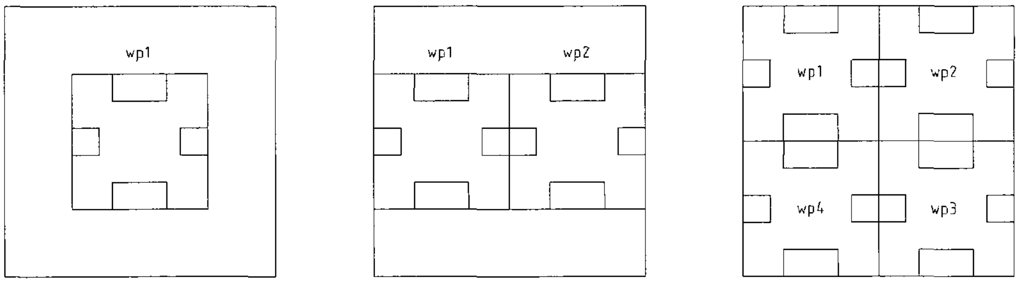
Figure 4.5: A processing path in CAPP World

4.2 The Manufacturing Cell

This section describes a simplified manufacturing cell environment in which the machining of workpieces takes place. The physical elements of this environment are:

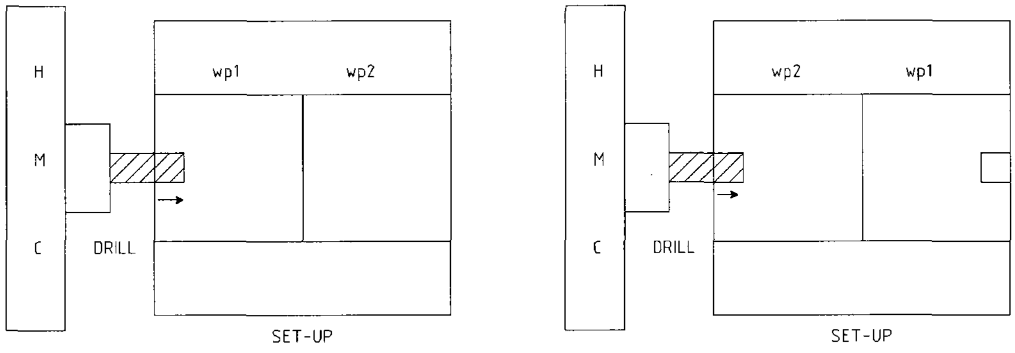
1. *Set-up stations.* These are places where workpieces are mounted and fixed on *pallets* to form *set-ups* for machining. In CAPP World a pallet is a square tablet whose edge is twice the workpiece edge; it can hold one, two, or four workpieces arranged as in Figure 4.6a, 4.6b, and 4.6c. (The central plane of the workpiece is always parallel to the pallet surface, so that a two-dimensional representation of a set-up suffices.) After the machining, the set-up is dissolved, and unfinished workpieces are combined into new set-ups for further processing. In general, a set-up exposes some workpiece features but may make others inaccessible. A part instance typically participates in several set-ups until its processing is completed.

2. *Horizontal machining centers (HMC).* A real-world manufacturing cell can contain various kinds of processing machinery. CAPP World is restricted to only one type, the HMC. An HMC has a horizontal *working table* that can rotate, where the pallet holding the set-up is installed, and a *cutting tool,* such as a drill, that moves with its axis in a horizontal position. A top view is shown in Figure 4.7.



(a) A pallet with one workpiece. (b) A pallet with two workpieces. (c) A pallet with four workpieces.

Figure 4.6: A set-up on a pallet with (a) one, (b) two, and (c) four workpieces.



(a) Drill workpiece 1. (b) Drill workpiece 2.

Figure 4.7: An HMC drilling a hole in a workpiece (a) and, after a table rotation, in another workpiece (b) within the same set-up.

Each machine in the cell is capable of performing certain *types of machining operations.* Since all features in the CAPP World are solid or hollow cylinders, HMC is restricted to two common operation types for such features, namely *drilling* and *milling.*

While working on a set-up, an HMC performs a sequence of machining operations at different positions and requiring different cutting tools. During its processing of the set-up, the HMC has access to a predefined *tool collection* (Figure 4.8). For each operation, it selects and mounts the appropriate tool, moves the spindle to the specified position, performs the cut, moves back to the original position, and then repeats the cycle for the next operation.

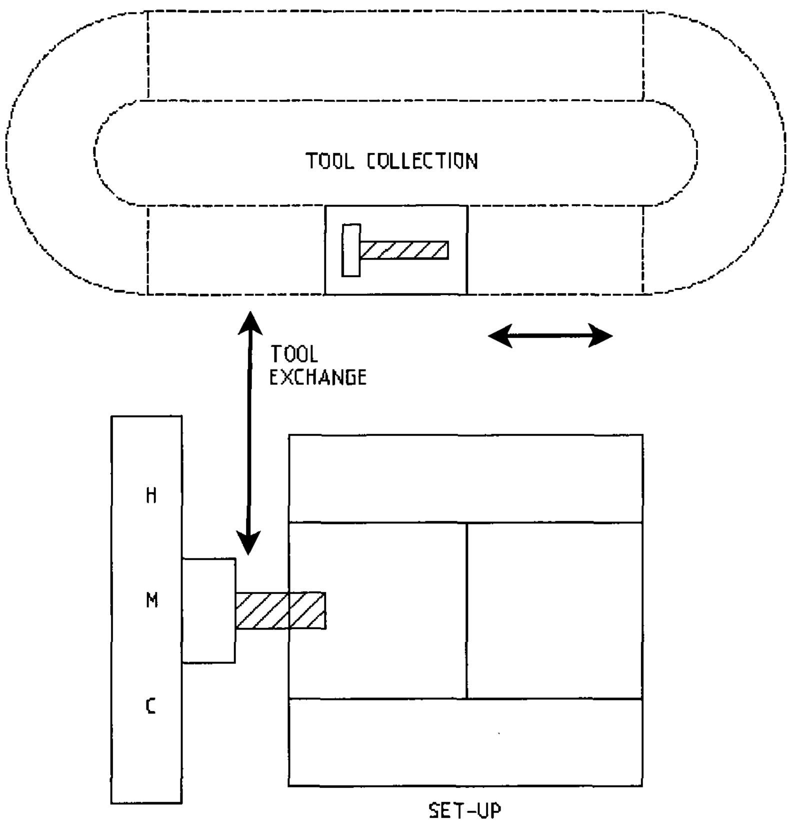


Figure 4.8: During set-up processing, an HMC exchanges tools from a prepared collection.

3. *Tool stations.* A tool station is a place where a tool collection is composed for a given set-up. The collection is then mounted on the HMC at the same time that the corresponding set-up is installed. The collection is removed when the set-up is completed (Figure 4.9). The set-up designer must ensure that the number of tools in the collection does not exceed the capacity of the turret carrying the tools.

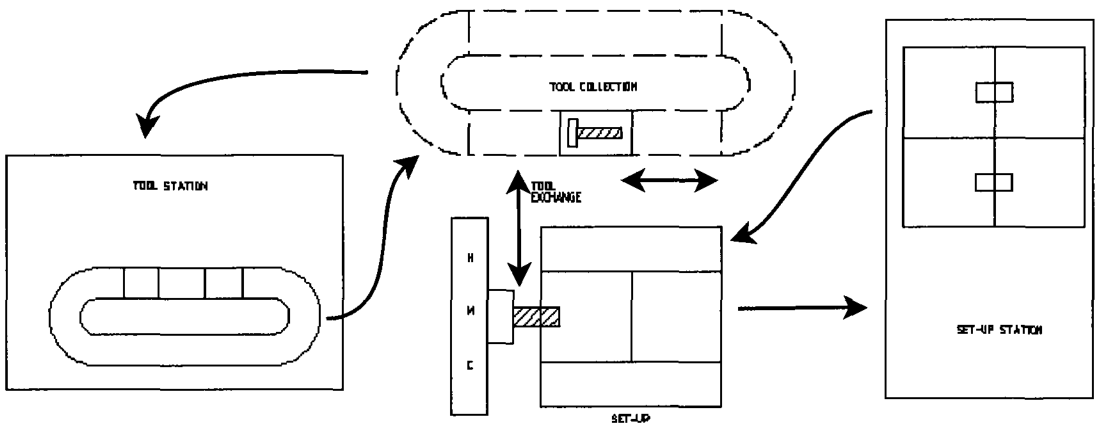


Figure 4.9: A new set-up and a new tool collection are installed simultaneously.

Note that any tools that were insufficiently used during the set-up processing are removed as well, linking the issues of set-up composition and efficient use of cutting tools. In general, the tool instances belong to a universe of available *tool types* for the specified operation types, that differ in size, quality, and cost.

The description of manufacturing cell does not address the issue of *how many* set-up stations, HMCs, and tool stations exist in the cell. This is because those numbers do not affect the construction of process plan. The sequencing of set-up processing on specific machines is the domain of shop-floor *scheduling*, a discipline related to but distinct from computer-aided process planning (CAPP). However, the process plan should facilitate efficient scheduling. For example, if the processing of each set-up instance takes approximately the same time, all machines can be reloaded in synch, which makes it is easier for the scheduler to avoid idle waits. Thus, the closeness of set-up processing times is one aspect of set-up plan quality.

4.3 Models of Process Plan Structures

In order to develop a process plan for a manufacturing task, one needs a precise description of what needs to be manufactured and with what resources. In general, a *process planning task* $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$ consists of the *assortment* $\mathcal{A} = \{A_1, \ldots, A_{n_\mathcal{A}}\}$, $n_\mathcal{A} \geq 1$, specifying the part series to be produced; the *configuration* $\mathcal{C}$ *of the production cell*, describing the available machinery; and the *requirements* $\mathcal{R}$, setting the performance objectives for the process plan. [2]In the case of CAPP World, we have discussed the assortment component in Section 4.1 and the cell component in Section 4.2. These two components completely specify the technical aspects of the manufacturing task that a process plan must unconditionally satisfy. The purpose of this section is to describe the structure and content of process plans in CAPP World, as well as the types of decisions that need to be made in order to develop a process plan for an assortment $\mathcal{A}$ and production cell $\mathcal{C}$. The third component of $\mathcal{T}$, the requirements $\mathcal{R}$, specifies the performance objectives such as the desired machining time and cost. $\mathcal{R}$ is related to the logistic and business aspects of the production process, but not to its technical feasibility and correctness. Its role in process plan development is discussed in the next section.

In general, a process plan can be viewed as a collection of related models that result from different types of specialized expert decisions. We briefly review each of those

[2]This and other basic definitions in process plan modeling have been adopted from Polajnar and Polajnar (2009) and elaborated here in the CAPP World context.

models as given in Polajnar et al. (2008b) and adapt them to the CAPP World context. They are:

1. *Part Machining Model (PMM).*

   This model represents the structure of individual parts and their machining processes. It is composed of four kinds of models, each representing a particular aspect of design expertise:

   (a) *Design Model (DM)* represents part design as supplied in the planning task. It specifies the *stock* from which the part of type P_i is produced, including its geometry and *part material*, as well as the part's geometry, surface finish, dimensional accuracy, and geometric tolerances (Halevi, 2003).

   In CAPP World, all parts are assumed to be of the same material, have the same stock geometry (cube of standard size), and have the part geometry described in terms of cylindrical features as explained in Section 4.1. For each design feature, the model includes information on which of the two cube sides that are parallel to the chosen central plane is used as the reference (datum) plane for the feature. All other information is left out of the core model, but can be selectively added as required by specific scenarios under study.

   (b) *Machining Feature Model (MFM)* represents the part type in terms of its set of machining features, their relationships, and resulting precedence relations. While machining feature recognition can be modeled in CAPP World, the current presentation leaves that aspect out and assumes that the machining feature decomposition has already been done. The precedence of machining features may result from geometric accessibility, but may also reflect purely machining concerns. For instance, a large hole may need to be done last for stability reasons.

(c) *Machining Operation Type Model (MOTM)* represents the processing of the part in terms of machining operation types. In general, a machining feature is realized through several operations. For each machining feature, the model specifies the general operation class, the types of its constituent machining operations, their precedence, and the possible *tool access directions (TAD)*. For each operation type it specifies the set of machines in the cell that can perform it.

In CAPP World, the only available general operation types are drilling and milling, and the only available machine type is horizontal machining center (HMC). MOTM contains information on the general operation class and its decomposition to constituent machining operation types. For instance, the machining of a cylindrical hole with drilling as the operation class can be carried out by three constituent operations: center drilling, drilling, and reaming, in that order of precedence. MOTM also includes the TAD values for the feature. Multiple TAD values are possible; for instance, a through hole feature could be accessed for machining from either side of the workpiece.

(d) *Machining Operation Method Model (MOMM)* represents the method by which each constituent operation is carried out. It specifies the *cutting tool* and the *cutting parameters* (such as length of cut, depth of cut, cutting speed, feed rate, tool life, and cutting forces). In real computer-aided process planning (CAPP), these choices are a source of major technological complexity.

In the core CAPP World, the choice of cutting tools is limited and simplified.

The parameters include the cutting time, speed, and tool life.

2. *Set-up Model (SM)*

   This model shows how workpieces are combined into set-ups. It contains the *set-up collection*, consisting of candidate set-up types, and the *set-up plan*, consisting of a partially ordered set of set-up instances formed from selected set-up types in the collection. The selection ensures that the entire assortment can be processed [3].

   In real CAPP, an important consideration in forming set-ups are the relationships between features that are induced by dimensional or geometric tolerances which use datum references. Such relationships may require that certain features be processed in the same set-up. In CAPP World, the axis of each cylindrical feature is in the central plane of the cube (which is horizontal during processing on HMC). The part design can indicate this fact by specifying the distance of the axis from either of the two sides of the cube that are parallel to the central plane. This datum reference plane will be the base on which the workpiece rests during machining. From datum references in the part design, the set-up designer must extract the *location direction* of each feature in order to know whether it can be included in a particular set-up.

   A *set-up type* is a sequence of workpiece roles that act as placeholders for workpieces to be processed; the positions in the sequence correspond to geometric placements on the pallet. For example, the set-up type corresponding to Figure 4.6c has four workpiece roles, $wp_1, \ldots, wp_4$. A workpiece role R consists of a pair of workpiece types: an *input* type W^{in} and an *output* type W^{out}. These are workpiece types of the same part type P_i, whose feature sets φ^{in} and φ^{out} are such that: (1)

[3] In real computer-aided process planning (CAPP), formation of set-ups involves the design of fixtures, which determines the range of cutting forces that can be applied. CAPP World leaves out fixture design, leading to a simplified set-up model.

$\varphi^{in} \supsetneq \varphi^{out}$; (2) each feature in $\Delta = \varphi^{in} - \varphi^{out}$ is geometrically accessible (based on its tool access direction and position on the pallet); and (3) each feature in Δ is technologically machinable (based on its location direction and position on the pallet). In other words, there is a path from the input workpiece type to the output workpiece type, corresponding to the machining (i.e., removal of machining features) that takes place in role R of the set-up type. The *feature set* of a set-up type is the union of Δ sets for all workpiece roles. A set-up type can be *instantiated* by assigning to each role R a workpiece instance of its input type W^{in} to produce a set-up instance. When the machining of the set-up instance is completed, the workpiece instance released from role R will be of its output type W^{out}.

The set-up collection is *complete* in the sense of including all set-up types that are needed for the processing of all part types in the assortment. In particular, this means that for each part type P_i there exists a sequence of workpiece types $w_1, \ldots, w_k$, $k \geq 1$, such that w_1 is the stock type (P_i, F_i), w_k is the finished part type $(P_i, \emptyset)$, and for each $j \in \{1, \ldots, k-1\}$ there is a set-up type in the collection that has a workpiece role with w_j as its input type and w_{j+1} as its output type. The condition ensures that parts of type P_i can be completely machined using set-up types in the collection. In order to provide alternative candidate options for set-up plan construction, the set-up collection normally ensures completeness through multiple alternative sequences of workpiece types.

The concepts of set-up type and set-up collection are defined in terms of part *types* in the assortment, without taking into account the series sizes. The remaining problem is to ensure that the set-up types in the collection can be used to form set-up instances that ensure that each part instance in the assortment can be machined to completion. This requirement is captured in the notion of set-up plan. Its definition in CAPP World is based on the simplifying assumption that each set-up instance is always completely processed on a single machine [4].

A *set-up plan* is a partially-ordered set of set-up instances formed from set-up types in the set-up collection in such a way that for each part instance in the assortment there is a path through workpiece instances in the set-up plan, with the following properties: (1) the path leads from the initial state (stock) to the final state (finished product) of the part instance; (2) if the path has a direct transition from a set-up instance s_1 to another set-up instance s_2, then $s_1 \leq s_2$ and the output type of workpiece instance in s_1 is identical to the input type of the workpiece instance in s_2 on the same path; and (3) the path is disjoint from the path of any other workpiece in the assortment.

3. *Process Plan Model (PPM)*

This model shows how the information from Machining Operation Type Model, Machining Operation Method Model, and Set-up Model is combined into a process plan.

For each set-up type S with feature set F, a *set-up program* of S is a sequence of machining operations for all features in F, whose order is consistent with the feature precedence and with the machining operation precedence. The set-up program is a higher-level representation of the numeric-code program that controls the HMC during the set-up processing by the shop-floor CAM system.

Finally, a *process plan* consists of a set-up plan, along with a set-up program for each set-up type participating in the plan. Note that the eventual processing order

[4]In real CAPP, the production cell may include several types of machines with different capabilities, leading to situations where a set-up instance may be processed in several steps on different machines. In CAPP World it is assumed that all machines in the cell belong to a single rather universal type, the HMC, and that this precludes the need to ever move a set-up instance between machines during processing.

of set-up instances in the production cell, which is determined by the shop-floor scheduler, must remain consistent with the partial order between set-up instances established in the set-up plan.

4.4 Actions in CAPP World

This section describes the actions that take place in CAPP World. These actions construct components of the models of process plan structures from the previous section. The presentation lists model components and explains how each is constructed. The order of presentation is consistent with the logical sequencing of steps in the construction of process plan. It is understood that each action is implemented using domain-specific heuristic that may rely on a knowledge base of specialist expertise, but the nature or structure of required knowledge is not represented. The presentation is meant to be entirely informal and to exclude possible bias towards any particular method of implementation. The "parameters" of each action are indices identifying the model component being constructed; it is understood that other information can be used in the construction process.

The starting assumptions are that a process planning task has been specified and that it includes a machining feature decomposition. This includes an assortment with n part types; we refer to them with the notation introduced in Section 4.3.

The Machining Operation Type Model

operation-class(part-type i, feature f)

Selects a general operation class, such as drilling, for a given feature.

tool-access-directions(part-type i, feature f)

Determines the axis and orientation of tool access direction (TAD) in the coordinate system of the part type. In CAPP World there is only one choice, except in the case of a through hole, where there are two possible TAD values.

machining-operations(part-type i, feature f)

Selects the types of machining operations that will be used in the machining of a given feature. Constructs a sequence $(mot_1(i, f), \ldots, mot_k(i, f))$ of $k \geq 1$ subcomponents, each specifying an operation type (e.g., center-drilling, drilling, or reaming) and the portion of the feature to be removed by the operation.

The Machining Operation Method Model

machining-operation-method(part-type i, feature f, mot m)

Selects the tool type and cost, and determines the cutting parameters for the machining operation type m. The cutting parameters depend on the tool selection and include: tool life, cutting time, and cutting speed.

The Set-up Model

part-type-priority-sequence

The analysis of part types in order to form set-up types usually begins with recognizing which parts types are critical to set-up formation and need to be analyzed first. This information is captured in the part type priority sequence. It is helpful in particular because the participation of different kinds of expertise in the design of set-up types requires careful organization in order to achieve efficiency.

location-direction(part-type i, feature f)

In general, datum references used by a feature in the part type design influence the part position in which the feature can be machined in a set-up. In CAPP World, the features can appear in four sides of the cube, while the "top" and "bottom" sides, i.e., the ones parallel to the central plane, can be used for datum reference. Location direction identifies which of those two planes has been used for datum reference for feature f in part type i.

feature-machining-summary(part-type i, feature f)

Extracts the information of interest for set-up formation from *MOTM* and *MOMM*. Specifically, it returns the tool types used in the machining of f, each with the total cutting time (possibly involving more than one operation) and tool life.

sides(part-type i)

Performs a partition of the feature set F_i into four subsets $(s_1(i), s_2(i), s_3(i), s_4(i))$, corresponding to the four machinable sides of the cube. Each feature is assigned to a particular side based on its TAD. In case of multiple TAD values, additional criteria can be used. In real CAPP, such criteria often involve dimensional and geometric tolerances not represented in core CAPP World. Criteria related to the machining times of the sides in a set-up can be used in our model.

side-machining-summary(part-type i)

Determines the totals of feature machining summaries for each pair (side, location direction) of part type i.

set-up-type-collection

Forms the set-up type collection for the set of all part types in the assortment. Each set-up type must meet the constraints of the production cell environment;

for instance, the number of tool instances used must not exceed the tool turret capacity. The construction of set-up type collection is both combinatorially and technologically complex. Various criteria can be used to identify desirable set-up types. Three examples of such criteria that are applicable in CAPP World are: (1) reducing the number of set-up instances; (2) improving the tool usage economy; and (3) equalizing set-up processing times. According to the first criterion, for instance, a solution in which two workpieces can be completely machined while combined in a single set-up is superior to the solution in which each workpiece is machined alone in a set-up. The second criterion favors grouping of workpieces that employ the same tool type, in order to reduce the relative impact of having to replace the last instance of that tool type before it was fully used up. The third criterion favors design choices that keep the processing times of set-up types in the collection close to each other in order to facilitate scheduling of the machines in the production cell. In more complex scenarios, set-up type construction may interact with machining operation design to achieve better outcomes in the process plan.

set-up-type-machining-summary(set-up-type s)

Determines the totals of side machining summaries for all workpiece sides exposed in set-up type s. This information can be used for evaluation of alternatives in the construction of set-up plan.

set-up-plan

Constructs a set-up plan from set-up types in the collection, using information on series size for each part type from the assortment. The action involves evaluation (again, using a variety of criteria) and comparison of alternatives provided by set-up type choices available in the collection.

The Process Plan Model

set-up-program(set-up-type s)

Constructs a set-up program for a given set-up type s. In CAPP World, the feature set of the set-up type into four subsets corresponding to different tool access directions. In operation sequencing, a transition between subsets requires pallet rotation. The partial ordering of features within part types and the ordering of machining operations realizing a particular feature together induce a partial ordering on the set of all machining operations in s. This action linearizes the ordering of machining operations, trying to reduce inter-operation times (for instance, by reducing the number of pallet rotations).

process-plan

Constructs a process plan from the set-up plan and the set-up programs of all set-up types participating in the plan. An initial version of the plan is constructed and evaluated with respect to the requirements stated in the Process Planning Task. Various strategies for revising portions of the plan can be used to produce further versions until the requirements are met or modified.

pp-assessment

Produces evaluations of various performance aspects of the current process plan version with respect to the requirements $\mathcal{R}$ stated in the process planning task.

pp-approval-status

Produces a decision on whether the current version of process plan is found to be satisfactory. This action sets the value of the predicate *pp-approved*, that is used to decide whether to construct a new version of the plan or terminate to planning activity with the current version as the result.

strategy

Formulates the strategy of process plan improvement in the case that the current version of process plan is not approved. The next version is constructed according to the stated strategy. The action formulates the strategy based on the assessment of the current version and may decide to retain an existing strategy.

Chapter 5

CAPP World as Framework for MAS Studies

In this chapter, we propose a series of scenarios for CAPP World. These scenarios are illustrated with pseudocode and Unified Modeling Language (UML) activity diagrams (Booch et al., 2005). Dumas and ter Hofstede (2001) discussed UML activity diagrams as a workflow specification language, and we apply their notation in our study. The pseudocode and UML activity diagrams used in the scenarios, and the mapping between pseudocode control structures and UML activity diagram elements are shown in Appendix A, with a summary of key elements in Table 5.1. The agent roles, actions, and other basic elements appearing in the pseudocode and diagrams have been given in Chapter 4. Agent encapsulation is presented in Section 5.1; agent communication and coordination mechanisms are presented in Section 5.2; a scenario on basic process planning construction is presented in Section 5.3; scenarios focusing on agents cooperation and coordination are presented in Section 5.4; some process plan improvement mechanisms are illustrated with scenarios presented in Section 5.5; global planning efficiency improvement is demonstrated by a scenario presented in Section 5.6; the issue on varying team composition is shown by a scenario presented in Section 5.7; and the issue on varying agents communication mechanisms is addressed in a scenario presented in Section 5.8.

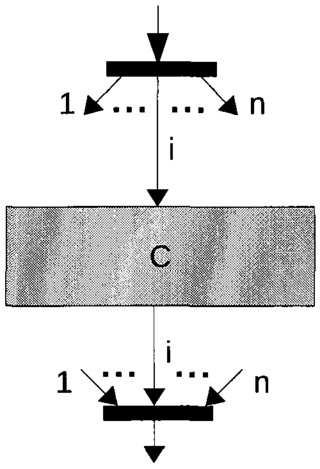
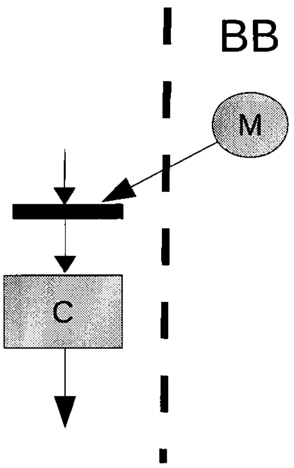
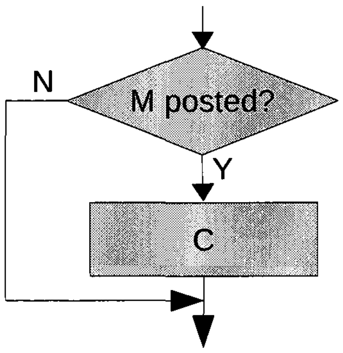
Pseudocode Language	**UML Activity Diagram Element**
foreach <*iterator*> **do** <*command*>	
when <*model component*> **posted** <*command*>	
if <*model component*> **posted**	

Table 5.1: The map of pseudocode and UML activity diagram (key elements).

5.1 Agent Encapsulation

The definition of CAPP World in Chapter 4 includes a description of models that need to be constructed in the development of a process plan, and a set of actions that need to be performed in order to construct the various components of those models. In order to define multiagent architectures operating in CAPP World, one needs to form a team of agents and assign each action to an agent. In other words, one must determine which part of computer-aided process planning (CAPP) functionality is *encapsulated* in a given agent. The encapsulation decisions impact the effectiveness of the multiagent architecture. Agent encapsulation has received substantial attention in recent surveys of multiagent systems for CAPP and was identified as an area for further study.

Methodologies for multiagent systems (MAS) development typically approach encapsulation by first defining the agent *roles*, then *instantiating* the roles, and then assigning the role instances to concrete agents (Zambonelli et al., 2003). A general role classification for the CAPP domain, proposed in Polajnar et al. (2008a), identifies five abstract roles: manager, designer, evaluator, strategist, and interagent. These abstract roles are not directly instantiated, but are used to derive other, more specialized roles, forming a class hierarchy. Roles in the hierarchy can be abstract (used only to derive other roles) or concrete (used for instantiation and possibly for derivation of more narrowly specialized roles). In team formation, instances of the same role can be assigned to multiple agents, e.g., for sharing high workload among identical agents that can operate in parallel. It is also possible to assign instances of different roles to the same agent, which then performs a combination of duties. The ancestry of all agent roles in the system eventually leads to the original set of five abstract roles.

Multiagent scenarios presented in this chapter follow the agent role classification outlined above, although the structure of CAPP World is in principle open to other approaches to encapsulation. However, within the adopted classification there remains a substantial latitude for varying how concrete roles are derived and how they are assigned to agents, resulting in different team compositions. These possibilities are discussed in Section 5.7. The scenarios in the rest of the chapter have a simple team structure consisting of five agents. We describe the responsibilities of each one in terms of models and actions of CAPP World as defined in Chapter 4.

Manager-Strategist (MS) This agent is the team leader. It performs two roles. As Manager, it provides the process planning task (*PPT*) to the team, reviews the assessment (provided by the Evaluator) of each version of process plan, and decides whether to approve it as the final result or to instruct the team to proceed to the next iteration. In the latter case, in its capacity of the Strategist, it formulates the strategy of process plan improvement for the team to follow (subject to approval by the Manager role). Note that the strategy does not literally tell the team members what to do. Instead, the agents autonomously combine the formulated strategy and their specialist expertise to synthesize individual criteria for their own specialist activities (e.g., MOD synthesizes the operation design criteria). The synthesis is envisioned as represented in the conceptual framework of belief-desire-intention (BDI) practical reasoning, but is not within the scope of this Thesis.

Machining Feature Designer (MFD)

This designer agent constructs the machining feature model (*MFM*) from the designs of all part types in the assortment. The agent requires both geometric and technological knowledge. As indicated earlier, we are not concerned with the details of machining feature recognition and do not discuss the internals of MFD.

Machining Operation Designer (MOD)

This designer agent requires in-depth knowledge of machining technology, which is one of the main sources of technological and combinatorial complexity in CAPP. For each part type, it first constructs the machining operation type model, *MOTM*, and then the machining operation method model, *MOMM*.

Set-up Designer (SD) This designer agent deals with substantial combinatorial, logical, and technological complexities while combining workpiece types into set-up types, developing the set-up collection, set-up plans, set-up programs, and finally integrating the process plan.

Evaluator (E)

This agent produces an assessment of a process plan so that the Manager can determine whether and how well it meets the requirements in $\mathcal{R}$ within the process planning task. The designer agents may also use the assessment in their formulation of the criteria to be used in the next iteration.

In relation to the CAPP role classification, the above team structure is derived from four out of the five abstract roles. Missing are the interagents, i.e., the interface agents to entities outside of the CAPP team. They have been left out in order to keep this presentation fully within the CAPP World framework. However, a CAPP World implementation could be made to interact with other systems, in which case interagents would be present. For instance, an enhancement that extends the tool selection to find concrete tools from a given set of vendors could have interagents interacting with the tool vendors' web sites, or possibly deployed to their host systems if the vendors were to provide such service. In the interest of simplicity, some of the other abstract roles have been treated as concrete and instantiated, as in the Evaluator case; the Manager-Strategist agent combines instances of two such roles. The Designer role has been used to derive three concrete roles, with instances assigned to MFD, MOD, and SD.

5.2 Communication and Coordination

Agents operating in CAPP World communicate with each other and coordinate their activities. While CAPP World is in principle open to any type of communication and coordination, the focus in this chapter is on *blackboard* mechanisms. The Blackboard (BB) is a shared space through which the agents exchange information and synchronize their activities. The computer-aided process planning (CAPP) model components, and the process plan itself, are placed on BB as they are completed, and the agents retrieve from BB the information they need. Specifically, agents interact with the Blackboard through the following mechanisms:

Subscribe

An agent indicates to BB that it wants to be notified when a particular type of model element is placed on BB. BB confirms receipt and sets up the requested notification mechanism.

Post

An agent places a model component on the blackboard, prompting delivery of notification messages to subscribers to that component type. The notification conveys what type of component was posted, but not the posted component itself.

Retrieve

An agent retrieves information from BB by executing a ***get*** command. The agent extracts only the information it needs through a mechanism such as remote method invocation. We do not assume specific implementation techniques which ensure that BB provides the necessary methods.

Publish

An agent places a model component on the blackboard, prompting delivery of notification messages to subscribers to that component type. The notification conveys both the type and the value of the component.

Notify

BB sends a message to subscribers indicating that a model component of a type to which they subscribe has been placed on BB. If the component was *posted*, the notification only conveys its type; if it was *published*, it also passes the value of the component. If an agent is internally multithreaded (which we normally assume) the notification message handler activates the thread scheduler, which can awaken a thread waiting for notification. This is the basic method of synchronization between agents used in the scenarios that follow. If a thread was waiting for published information, it is normally awakened and given the value passed in the notification. However, it is also possible that the information in the message is intended for the thread scheduler. An example of this is shown in Scenario 2b.

Examples of multiagent scenarios in the sections followed are in some cases illustrated with pseudocode and with UML (Unified Modeling Language) activity diagrams. In addition to conventional constructs such as *while* loops, *for* loops, conditionals, etc., the pseudocode uses commands that are externally synchronized with BB notifications. The command of the form "**when** m **posted** C" blocks until a component of type m has been posted and then executes C. The agent can then retrieve information from the component using ***get***. The command "**when** m **published** C" is similar except that no subsequent ***get*** is needed since the entire component value has already been placed in the variable m. The non-blocking versions of these commands are "**if** m **posted** C_1 [**else** C_2]" and "**if** m **published** C_1 [**else** C_2]", where the part in brackets is optional. Here the agent executes C_1 if m has been posted or published; otherwise it proceeds, or executes C_2 if the "else" branch exists. Another construct of interest is "**foreach** $i \in S$ **do** C", which is a multithreaded loop, whose iterations execute concurrently (normally on the same processor). The iterator S can be a set, in which the choice of which thread to schedule to run is non-deterministic, or it can be a linear sequence, in which case the scheduler treats it as a priority list. The representation of these key constructs in activity diagrams is shown in Table 5.1.

5.3 Scenario 1: Basic Process Plan Construction

Scenario 1 illustrates the construction of the initial version of process plan for a given process plan task (*PPT*) by the Set-up Designer (SD) and the Machining Operation Designer (MOD) agents. *PPT* is defined as $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$, where $\mathcal{A} = (A_1, \ldots, A_n)$ is the *assortment* of $n \geq 1$ *series* A_i, each consisting of a part design type P_i, and the number of n_i of instances to be produced; $\mathcal{C}$ is the *configuration of the production cell*, describing the available machines; and $\mathcal{R}$ represents the *requirements* that specify the performance objectives for the process plan. In general, the information in $\mathcal{T}$ is used to construct a process plan that describes in detail how the parts specified in $\mathcal{A}$ can be manufactured by the machines in the production cell $\mathcal{C}$ to meet the requirements $\mathcal{R}$. The designer agents communicate and coordinate their actions through the blackboard (BB) using posting and retrieval mechanisms.

In Scenario 1, the Manager-Strategist (MS) initially posts *PPT* on BB. SD and MOD retrieve the requirements for the task ($\mathcal{R}$) and form their individual expert criteria. Next, the Machining Feature Designer (MFD) agent posts the Machining Feature Model (MFM_i) for each part type i. SD and MOD retrieve information from MFM_i pertaining to their specific roles. After that, MOD develops and posts on BB the Machining Operation

Model (MOM_i) for each part type i (this includes both operation types and methods). For part type i SD constructs side k (for $k = 1, \ldots, 4$) using CAPP action *make-side(i,k)*[1]. SD then constructs the set-up types, uses information on series sizes to form the set-up plan, develops set-up programs for all set-up types, and finally constructs and posts the process plan.

The pseudocode description of agents SD and MOD are shown in Figure 5.1 (SD) and Figure 5.2 (MOD) respectively, and the activity diagram in Figure 5.3.

[1]This action uses information about location direction and tool access directions to determine which machining features belong to which side of the workpiece. The descriptions of this and other CAPP actions are given in the Chapter 4.

```
agent Set-up Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        for $i \in \{1,\ldots,n\}$ do
            get BB.series-size(i) from $\mathcal{A}$;
        get BB.set-up-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make set-up-design-criteria;
    }
    foreach $i \in \{1\ldots n\}$ do {
        when $MFM_i$ posted {
            get $F_i$ =BB.feature-id-set($MFM_i$);
            for $f \in F_i$ do
                get BB.location-direction(f);
        }
        when $MOM_i$ posted
            for $f \in F_i$ do {
                get BB.operation-class(f);
                get BB.tool-access-directions(f);
                get BB.feature-machining-summary(f);
            }
        for $k \in \{1,\ldots,4\}$ do {
            make side(i,k);
            make side-machining-summary(i,k);
        }
    }
    make set-up-types;
    make set-up-plan;
    make set-up-programs;
    make process-plan;
    post BB.process-plan;
}
```

Figure 5.1: Scenario 1: The Set-up Designer (SD).

```
agent Machining Operation Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        get BB.operation-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make operation-design-criteria;
    }
    foreach $i \in \{1 \ldots n\}$ do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.feature-description(f);
            make MOM(f);
        }
        make $MOM_i$;
        post BB.$MOM_i$;
    }
}
```

Figure 5.2: Scenario 1: The Machining Operation Designer (MOD).

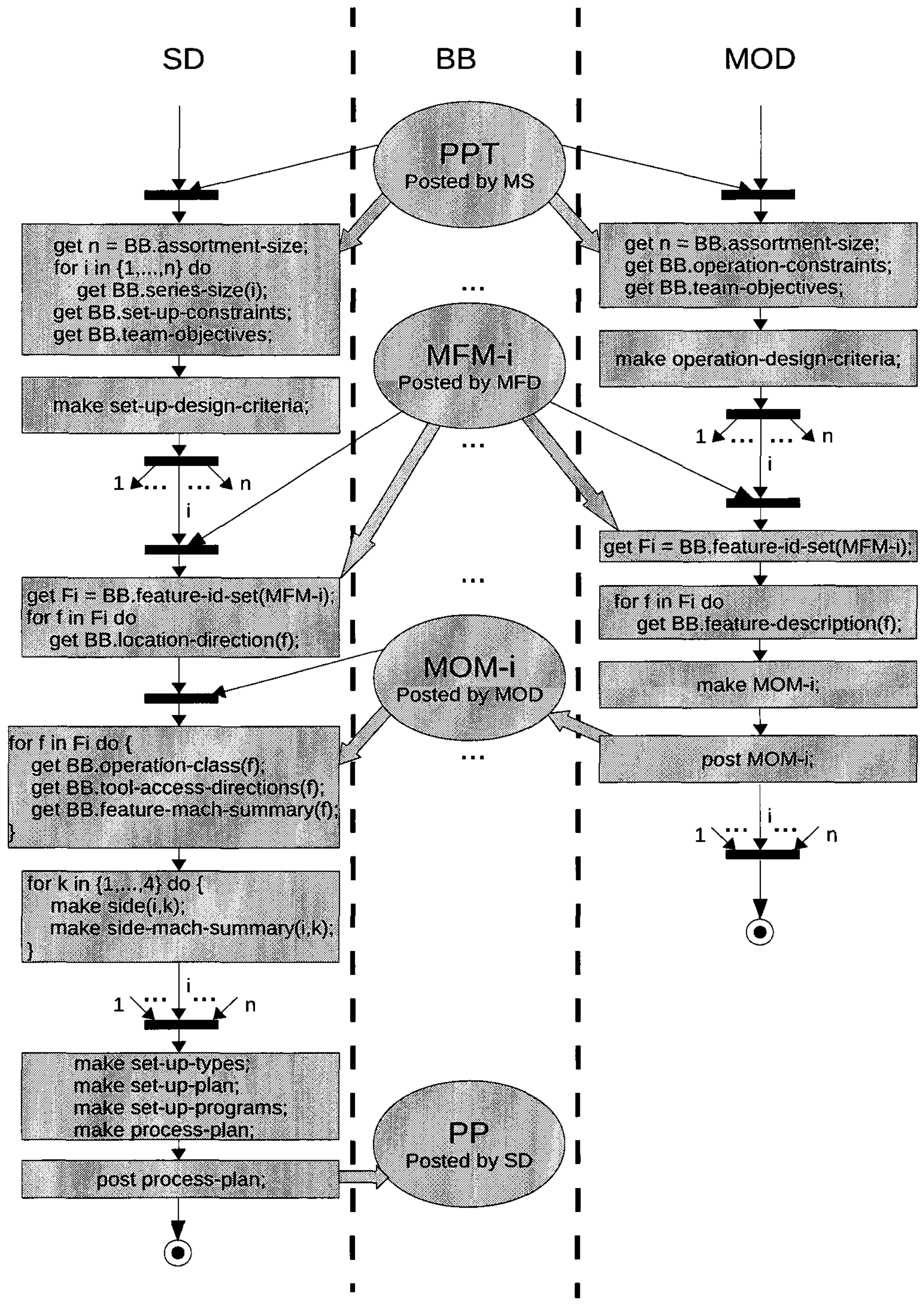

SD
BB
MOD
PPT
Posted by MS
get n = BB.assortment-size;
for i in {1,...,n} do
get BB.series-size(i);
get BB.set-up-constraints;
get BB.team-objectives;
get n = BB.assortment-size;
get BB.operation-constraints;
get BB.team-objectives;
MFM-i
Posted by MFD
make set-up-design-criteria;
make operation-design-criteria;
1 ... ... n
i
get Fi = BB.feature-id-set(MFM-i);
get Fi = BB.feature-id-set(MFM-i);
for f in Fi do
get BB.location-direction(f);
for f in Fi do
get BB.feature-description(f);
MOM-i
Posted by MOD
make MOM-i;
post MOM-i;
for f in Fi do {
get BB.operation-class(f);
get BB.tool-access-directions(f);
get BB.feature-mach-summary(f);
}
for k in {1,...,4} do {
make side(i,k);
make side-mach-summary(i,k);
}
make set-up-types;
make set-up-plan;
make set-up-programs;
make process-plan;
PP
Posted by SD
post process-plan;

Figure 5.3: Scenario 1: Basic Process Plan Construction

5.4 Scenarios 2a, 2b: Cooperation and Coordination

5.4.1 Scenario 2a

Scenario 2a seeks to improve the efficiency of process planning through better coordination between the Set-up Designer (SD) and the Machining Operation Designer (MOD) in comparison with Scenario 1. The idea is to reduce the waiting of SD for MOD's results in two ways. First, MOD posts the machining operation type model for each part i ($MOTM_i$) as soon as it is generated, allowing SD to proceed with forming of sides for part type i while MOD works on other part types to produce their $MOTM$ and later the machining operation method model, $MOMM$. Second, MOD constructs the machining models for part types in the order of a part type priority sequence supplied by SD. This ensures that SD gets machining information in the order needed for its set-up construction. A quantitative experimental study in CAPP World could indicate how significant improvements in the efficiency of process planning result from such improved coordination between agents.

As in Scenario 1, the Manager-Strategist (MS) initially posts on BB the process planning task (PPT) of the form $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$. Next, the Machining Feature Designer (MFD) agent posts the machining feature model (MFM_i) for each part type i. SD and MOD retrieve the required information from MFM_i. SD makes and posts the part type priority sequence indicating the priorities of part types. In the order of a part type priority sequence supplied by SD, MOD constructs the machining operation type model ($MOTM_i$) for part type i and posts it on BB. SD uses this information and information retrieved from MFM_i to construct sides for the part type i in the order of the part type priority sequence. At the same time, for every $MOTM$ of part type i, MOD is constructing the machining operation method model $MOMM$. When $MOMM_i$ is constructed and posted on BB, SD retrieves the *side-machining-summary*, which contains the total cutting time for all features on the side and the list of tools used for the side, the cutting time of each machining operation, and tool life for each tool. SD uses *side-machining-summary* to create set-up type collection, employs information on series sizes to form the set-up plan, develop set-up programs for all set-up types, and finally construct and post the process plan.

The pseudocodes for SD and MOD is shown in Figure 5.4 and Figure 5.5 respectively; and Figure 5.6 shows the activity diagram of this scenario.

```
agent Set-up Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        for $i \in \{1, \ldots, n\}$ do
            get BB.series-size(i) from $\mathcal{A}$;
        get BB.set-up-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make set-up-design-criteria;
    }
    foreach $i \in \{1 \ldots n\}$ do {
        when $MFM_i$ posted {
            get $F_i$ =BB.feature-id-set($MFM_i$);
            for $f \in F_i$ do {
                get BB.location-direction(f);
                get BB.prioritization-info(f);
            }
        }
    }
    make ptps = part-type-priority-sequence;
    post BB.part-type-priority-sequence;
    foreach $i \in$ ptps do {
        when $MOTM_i$ posted {
            for $f \in F_i$ do {
                get BB.operation-class(f);
                get BB.tool-access-directions(f);
            }
            for $k \in \{1, \ldots, 4\}$ do
                make side(i,k);
        }
    }
    foreach $i \in$ ptps do {
        when $MOMM_i$ posted {
            for $f \in F_i$ do
                get BB.feature-machining-summary(f);
            for $k \in \{1, \ldots, 4\}$ do
                make side-machining-summary(i,k);
        }
    }
    make set-up-types;
    make set-up-plan;
    make set-up-programs;
    make process-plan;
    post BB.process-plan;
}
```

Figure 5.4: Scenario 2a: The Set-up Designer (SD).

```
agent Machine Operation Designer {
    when $PPT$ posted {
        get $n$ = BB.assortment-size from $\mathcal{A}$;
        get BB.operation-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make operation-design-criteria;
    }
    when part-type-priority-sequence posted
        get ptps = BB.part-type-priority-sequence;
    foreach $i \in$ ptps do {
        when $MFM_i$ posted
            get $F_i$ = BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.feature-description(f);
            make MOTM(f);
        }
        make $MOTM_i$;
        post BB.$MOTM_i$;
    }
    foreach $i \in$ ptps do {
        for $f \in F_i$ do
            make MOMM(f);
        make $MOMM_i$;
        post BB.$MOMM_i$;
    }
}
```

Figure 5.5: Scenario 2a: The Machining Operation Designer (MOD).

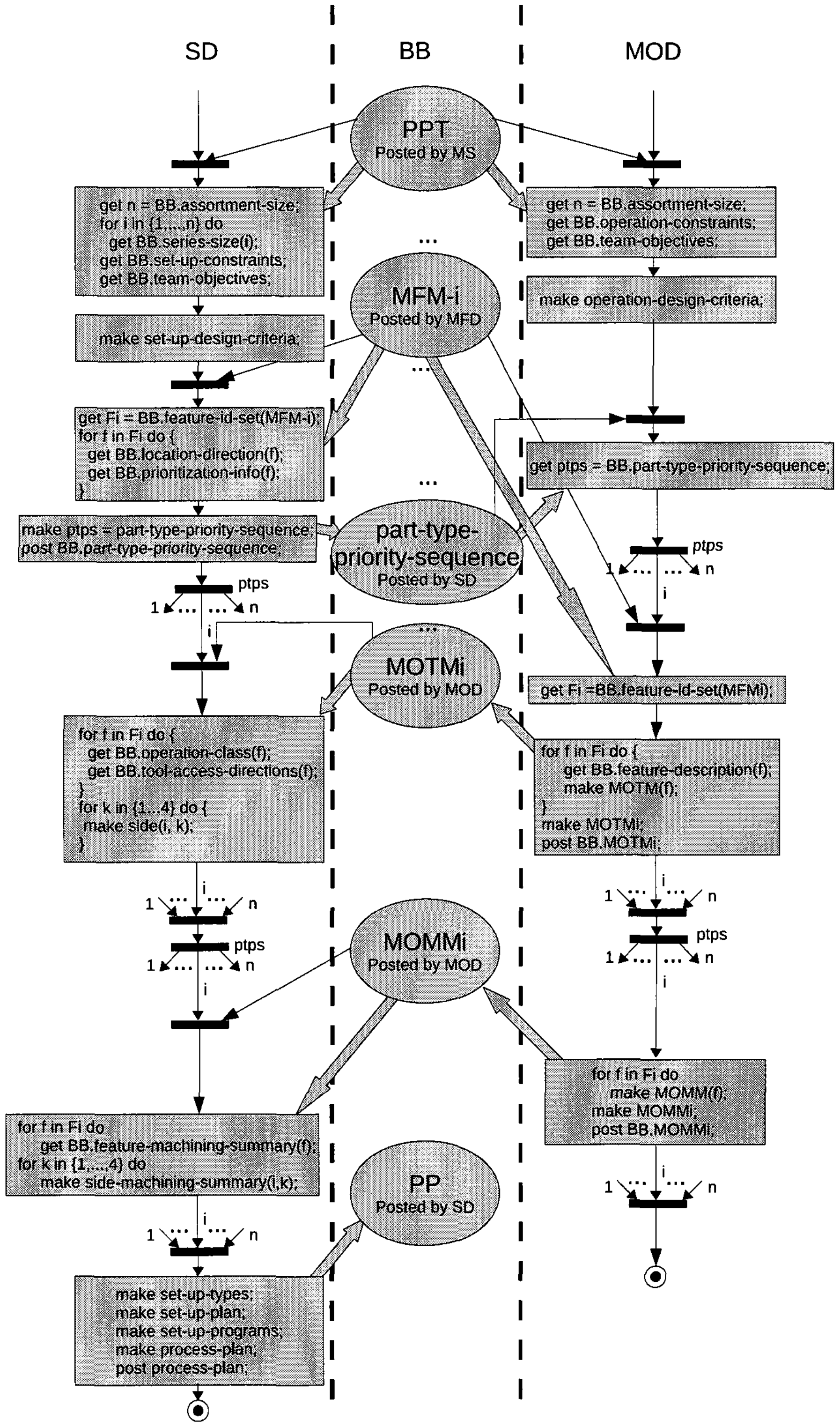

SD
BB
MOD
PPT
Posted by MS
get n = BB.assortment-size;
for i in {1,...,n} do
get BB.series-size(i);
get BB.set-up-constraints;
get BB.team-objectives;
get n = BB.assortment-size;
get BB.operation-constraints;
get BB.team-objectives;
make set-up-design-criteria;
MFM-i
Posted by MFD
make operation-design-criteria;
get Fi = BB.feature-id-set(MFM-i);
for f in Fi do {
get BB.location-direction(f);
get BB.prioritization-info(f);
}
get ptps = BB.part-type-priority-sequence;
make ptps = part-type-priority-sequence;
post BB.part-type-priority-sequence;
part-type-priority-sequence
Posted by SD
ptps
1
n
i
MOTMi
Posted by MOD
get Fi =BB.feature-id-set(MFMi);
for f in Fi do {
get BB.operation-class(f);
get BB.tool-access-directions(f);
}
for k in {1...4} do {
make side(i, k);
}
for f in Fi do {
get BB.feature-description(f);
make MOTM(f);
}
make MOTMi;
post BB.MOTMi;
MOMMi
Posted by MOD
for f in Fi do
make MOMM(f);
make MOMMi;
post BB.MOMMi;
for f in Fi do
get BB.feature-machining-summary(f);
for k in {1,...,4} do
make side-machining-summary(i,k);
PP
Posted by SD
make set-up-types;
make set-up-plan;
make set-up-programs;
make process-plan;
post process-plan;

Figure 5.6: The Activity Diagram of Scenario 2a.

5.4.2 Scenario 2b

Scenario 2a reduces the waiting time of the Set-up Designer (SD) for the results posted by the Machining Operation Designer (MOD) compared to Scenario 1. However, the modified solution also introduces new waiting: MOD cannot start to develop the machining models until SD has posted the part type priority sequence in which the machining models for the part types ought to be constructed. A more natural coordination of activities in the team would be for MOD to start and keep working on part types in any order until SD's priorities have been defined. Upon being informed of the priorities, MOD can start scheduling threads corresponding to different part types accordingly. This approach is realized in Scenario 2b. It certainly fits better with the idea of agents as autonomous, intelligent, and proactive entities.

Scenario 2b uses some additional mechanisms for communication and thread scheduling. First, note that SD now announces the part type priority sequence by a ***publish*** rather than ***post*** command. The effect of either command is that the announced component is placed on BB and the subscribers receive a notification message. The difference is that in the case of ***publish*** the message contains the value of the entire published component, while in the case of ***post*** it only informs about the type of component that has been posted and can be retrieved by a ***get*** command. A published component's value becomes known to the subscribing agent as soon as the agent receives the notification.

At the receiving end, MOD executes the construct

foreach $i \in \{1 \dots n\}$ **by** *part-type-priority-sequence* **do** $\{\dots\}$

The recipient of the published priority information in this case is the thread scheduler controlling the execution of the **foreach** loop. Before the priority sequence has been published, the scheduling is non-deterministic; once the priority sequence is published, the scheduler observes the priorities specified in the sequence. The mechanism thus provides a more efficient form of coordination between SD and MOD, removing the deficiency in Scenario 2a. The pseudocode fragments of the two agents in Figure 5.7 (SD) and Figure 5.8 (MOD) show this interaction in context.

It should be noted that the control of one agent (SD) over the internal thread scheduler of another agent (MOD) is not automatic (as in this simplified example), because such design would violate the individual autonomy of agents. The receiving agent should be allowed to decide on whether to apply the received scheduling priorities, but such deliberation should be efficient in order to not compromise the efficiency of thread scheduling.

```
agent Set-up Designer {
        ...
    make ptps = part-type-priority-sequence;
    publish BB.part-type-priority-sequence;
    foreach $i \in ptps$ do {
        when $MOTM_i$ posted
            for $f \in F_i$ do {
                get BB.operation-class(f);
                get BB.tool-access-directions(f);
            }
        for $k \in \{1,\ldots,4\}$ do
            make side(i,k);
    }
        ...
}
```

Figure 5.7: Scenario 2b: The Set-up Designer (SD).

```
agent Machining Operation Designer {
        ...
    foreach $i \in \{1 \ldots n\}$ by part-type-priority-sequence do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.feature-description(f);
            make MOTM(f);
        }
        make$MOTM_i$;
        post BB.$MOTM_i$;
    }
        ...
}
```

Figure 5.8: Scenario 2b: The Machining Operation Designer (MOD).

5.5 Scenarios 3a, 3b: Process Plan Improvement

5.5.1 Scenario 3a

Scenario 3a illustrates the process plan improvement through changes in the set-up design based on the evaluation assessment of the process plan and the improvement strategy. The assessment is done by the Evaluator (E) agent and the improvement strategy by the Manager-Strategist (MS) agent.

In this scenario, MS initially posts on BB the process planning task (PPT) of the form $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$. The Evaluator (E), the Set-up Designer (SD), and the Machining Operation Designer (MOD) retrieve the requirements for the task ($\mathcal{R}$) and form their individual expert criteria. SD initially constructs a process plan, using information posted by the Machining Feature Designer (MFD) and MOD, as in previous scenarios.

MOD is identical as in Scenario 1 (please see Figure 5.2). When SD has posted an initial process plan, E evaluates it and posts an assessment on BB. MS retrieves the assessment, uses it to decide on the approval of the process plan, and posts the approval status. If the approval is granted, everyone terminates the process plan construction; if not, MS formulates a process plan improvement strategy and posts it on BB. Other agents retrieve the strategy; agents that participate in the revision, in this case SD only, retrieve the assessment as well. SD then uses the assessment and the strategy to adjust its criteria for set-up type collection design, constructs a new version of process plan and posts it on BB.

The pseudocode descriptions of agents MS, E, and SD are shown in Figures 5.9 (MS), Figure 5.10 (E), and Figure 5.11 (SD) respectively, and the activity diagram in Figure 5.12. The pseudocode description of MOD is not included (it is identical to MOD in Figure 5.2), and it is not shown in the activity diagram. In Figure 5.11 and Figure 5.12, the actions of SD to make an initial process plan are as identical as in Scenario 1 (please see Figure 5.1 and Figure 5.3), denoted here as: [**construction of initial process plan**].

```
agent Manager-Strategist {
    make process-planning-task;
    post process-planning-task;
    pp-approved = false;
    while not pp-approved do {
        when pp-assessment posted
            get BB.pp-assessment;
        make pp-approval-status;
        post pp-approval-status;
        if not pp-approved do {
            make strategy;
            post BB.strategy;
        }
    }
}
```

Figure 5.9: Scenario 3a: The Manager-Strategist (MS).

```
agent Evaluator {
    when PPT posted {
        get BB.PPT;
        make evaluation-criteria;
    }
    pp-approved = false;
    while not pp-approved do {
        when PP posted
            get BB.pp-evaluation-info;
        make pp-assessment;
        post pp-assessment;
        when pp-approval-status posted
            get BB.pp-approval-status;
        if not pp-approved do {
            when strategy posted
                get BB.strategy;
            make evaluation-criteria;
        }
    }
}
```

Figure 5.10: Scenario 3a: The Evaluator (E).

```
agent Set-up Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        for $i \in \{1,\ldots,n\}$ do
            get BB.series-size(i) from $\mathcal{A}$;
        get BB.set-up-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make set-up-design-criteria;
    }
    [construction of initial process plan;]
    pp-approved = false;
    while not pp-approved do {
        post process-plan;
        when pp-approval-status posted
            get BB.pp-approval-status;
        if not pp-approved do {
            when strategy posted
                get BB.strategy;
            get BB.pp-assessment;
            make set-up-design-criteria;
            foreach $i \in \{1\ldots n\}$ do
                for $k \in \{1,\ldots,4\}$ do {
                    make side(i,k);
                    make side-machining-summary(i,k);
                }
            make set-up-types;
            make set-up-plan;
            make set-up-programs;
            make process-plan;
        }
    }
}
```

Figure 5.11: Scenario 3a: The Set-up Designer (SD).

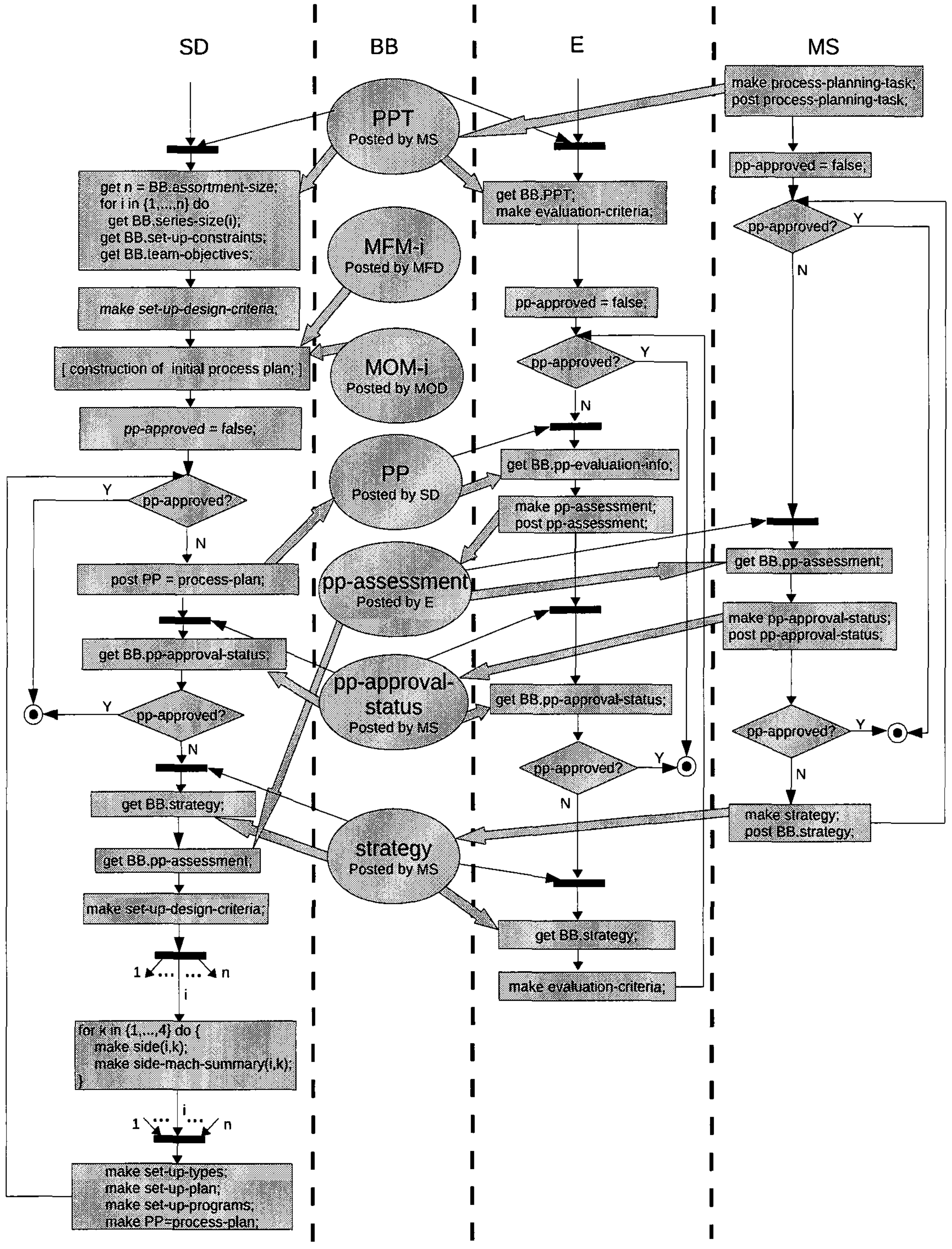


Figure 5.12: The Activity Diagram of Scenario 3a.

5.5.2 Scenario 3b

The previous scenario (i.e. Scenario 3a) illustrates the process plan improvement through changes in the set-up design based on the evaluation assessment of the process plan and the improvement strategy. Scenario 3b illustrates the process plan improvement through changes in the design of Machining Operation Method Model (*MOMM*) by the Machining Operation Designer (MOD) based on the evaluation assessment of the process plan and the improvement strategy. The Set-up Designer (SD) is similar as in Scenario 1 (please see Figure 5.2), except that it now checks the approval status and constructs new versions of process plan as necessary.

In Scenario 3b, the Manager-Strategist (MS) initially posts on BB the process planning task (*PPT*) of the form $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$. The Evaluator (E), SD, and MOD retrieve the requirements for the task ($\mathcal{R}$) and prepare their individual expert the criteria. SD initially constructs a process plan, using information from the Machining Feature Designer (MFD) and MOD agents, as in previous scenarios.

When SD has posted an initial process plan, E evaluates it and posts an assessment on BB. MS retrieves the evaluation, uses it to decide on the approval of the process plan, and posts the approval status. If the approval is granted, everyone terminates the process plan construction; if not, MS formulates a process plan improvement strategy and posts it on BB. Other agents retrieve the strategy; designer agents that participate in the revision, in this case MOD only, also retrieve the assessment. MOD then uses the assessment and the strategy to adjust its criteria for *MOMM* design and posts new designs on BB. The other agents proceed to construct and evaluate a new version of the process plan.

The pseudocode description of agents MS, E, and MOD are shown in Figure 5.13, 5.14, and 5.15 respectively, and the activity diagram in Figure 5.16. The pseudocode description of SD is not included (it is iterative version of SD in Figure 5.1) and it is not shown in the activity diagram.

```
agent Manager-Strategist {
    make process-planning-task;
    post process-planning-task;
    pp-approved = false;
    while not pp-approved do {
        when pp-assessment posted
            get BB.pp-assessment;
        make pp-approval-status;
        publish pp-approval-status;
        if not pp-approved do {
            make strategy;
            post BB.strategy;
        }
    }
}
```

Figure 5.13: Scenario 3b: The Manager-Strategist (MS).

```
agent Evaluator {
    when PPT posted {
        get BB.ppt-info;
        make evaluation-criteria;
    }
    pp-approved = false;
    while not pp-approved do {
        when PP posted
            get BB.pp-evaluation-info;
        make pp-assessment;
        post pp-assessment;
        when pp-approval-status published
            if not pp-approved do {
                when strategy posted
                    get BB.strategy-info;
                make evaluation-criteria;
            }
    }
}
```

Figure 5.14: Scenario 3b: The Evaluator (E).

```
agent Machine Operation Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        get BB.operation-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make operation-design-criteria;
    }
    foreach $i \in \{1 \ldots n\}$ do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.feature-description(f);
            make MOTM(f);
        }
        make $MOTM_i$;
        post BB.$MOTM_i$;
    }
    pp-approved = false;
    while not pp-approved do {
        foreach $i \in \{1 \ldots n\}$ do {
            for $f \in F_i$ do
                make MOMM(f);
            make $MOMM_i$;
            post BB.$MOMM_i$;
        }
        when pp-approval-status published
            if not pp-approved do
                when strategy posted
                    get BB.strategy;
                get BB.assessment;
                make operation-design-criteria;
    }
}
```

Figure 5.15: Scenario 3b: The Machining Operation Designer (MOD).

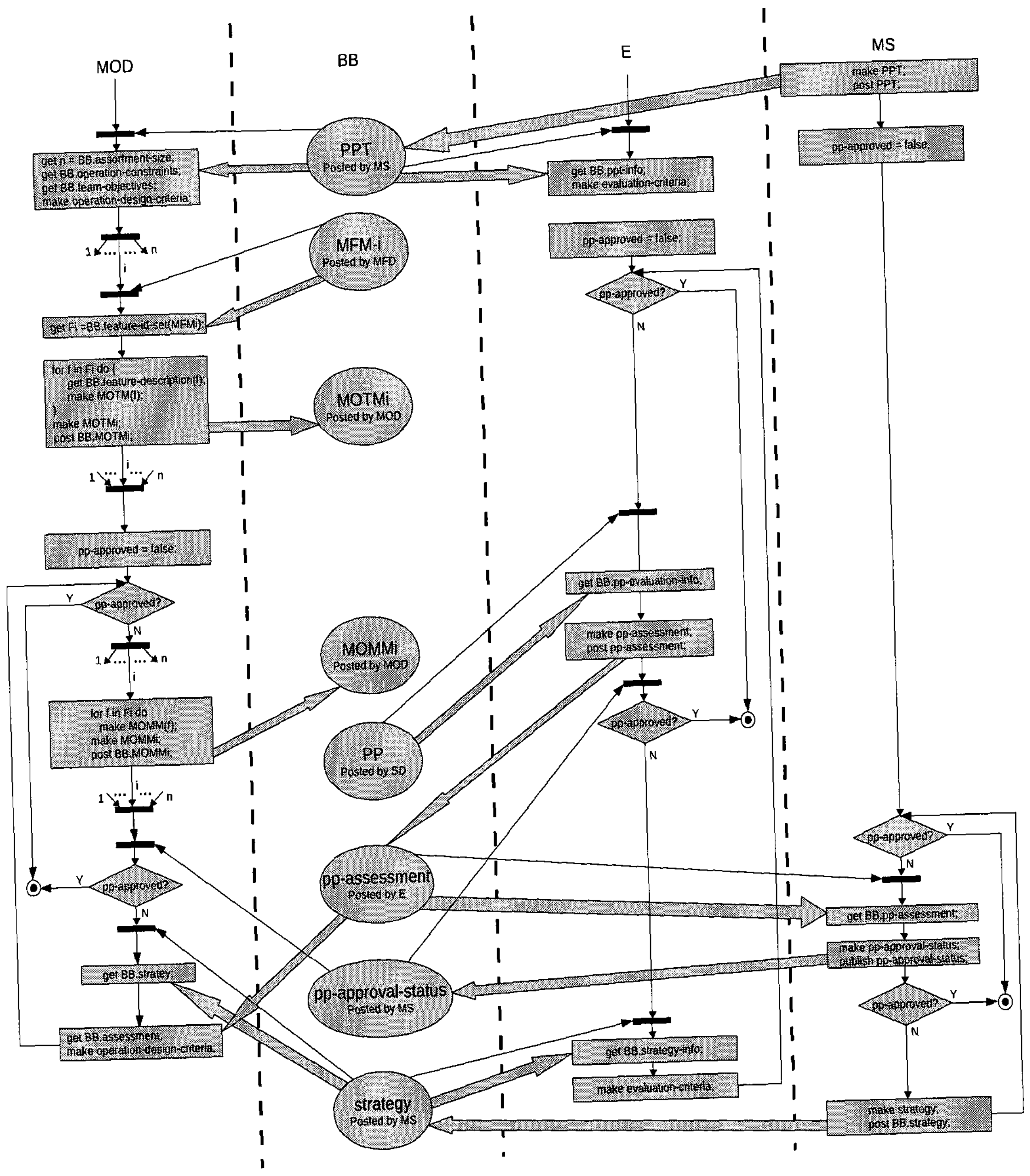


Figure 5.16: The Activity Diagram of Scenario 3b.

5.6 Scenario 4: Improving Efficiency of Planning

This scenario illustrates the improvement of planning process by caching a limited number of intermediate decision choices in each of the process planning steps. The sets of decision choices are sometimes results of extensive and time consuming computing, and by caching them in the individual space, agent may save substantial time in case of backtracking or iterative improvements of the process plan. Designer agent, such as machining operation designer, makes particular decision within its local scope to the best of its knowledge and it is not aware of the global consequences of the decision. The assessment of the achieved goal or sub-goals may indicate that the previous decisions have to be corrected. In that situation a significant amount of computation can be saved if an adequate solution has previously been cached by the designer agent. As in other forms of caching, it is a research problem to determine good caching strategies that make such outcomes highly probable.

A simple example illustrating the use of caching in the core CAPP World can be based on Scenario 3b, that shows iterative modifications of process plan aimed at meeting the performance requirements through improvements in the design of machining operation methods, i.e., the selection of cutting tools and parameters. Caching is introduced into design decisions of the Machining Operations Designer (MOD), as it constructs iterations of the machining operation method model, *MOMM*, for the initially constructed machining operation type model, *MOTM*. Namely, in the first iteration of process plan construction, MOD will cache a small set of valid solutions for *MOMM* in its individual space, and post on BB the one that seems to fit best the operation design criteria currently observed by MOD. When the construction of the initial version of process plan has been completed, and the Set-up Designer (SD) has posted it on BB, the Evaluator (E) posts an assessment of the current version of process plan, and the Manager-Strategist

(MS) decides whether further improvement is necessary. Given the complexity of process planning, it is normally expected that more than one iteration will be needed to meet the performance requirements. If the current version is not approved, MS uses the assessment to formulate an improvement strategy. Based on the posted strategy and its own expertise, MOD then modifies its own operation design criteria. In the case of a good caching strategy, it is likely that one of the cached alternative designs fits the new criteria, saving the computation that would be needed to construct it from scratch.

Scenario 4 could be used for comparative studies for effectiveness of caching strategies in operation method design. In order to demonstrate significant differences in the efficiency of process planning between the two scenarios, domain-specific caching strategies would normally be investigated in a framework consisting of the core CAPP World enhanced with a particular type of technological complexity. For instance, one could widen the decision space for cutting tool and parameter selection in order to increase the relevance to real CAPP. In order to produce practically useful simulation studies of effectiveness of caching, the timing metrics ought to realistically reflect the designer agent's deliberation time in deciding what to cache, the time to generate a solution in real CAPP, and the time to retrieve a solution from the cache; the balancing of hit ratio vs. cache size also needs to be considered.

5.7 Scenario 5: Varying Team Composition

Scenario 5 is variation of Scenario 2a, but uses a different composition of the multiagent team. In this scenario, machining operation design is performed by two cooperating agents: Machining Operation Type Designer (MOTD) and Machining Operation Method Designer (MOMD). For each part type i, MOTD designs its machining operation type model ($MOTM_i$), while MOMD is responsible for designing its machining operation method model ($MOMM_i$).

Initially, designer agents SD, MOTD, and MOMD collect the necessary information from the process planning task (PPT) and make their own design criteria. After the Machining Feature Designer (MFD) posts the Machining Feature Model (MFM_i) for a part type i, all three designer agents extract the feature information they need. Next, SD posts the priorities of part types for the machining operation designers to work on. MOTD constructs $MOTM_i$ for each part type i and posts it on BB, in the order of priority specified by SD. As soon as $MOTM_i$ is posted, the other two designers can use it—SD to identify the sides of the part type, and MOMD to design the machining methods for its operations. In particular, for every feature $f \in F_i$, MOMD retrieves from BB the set OT_f of machining operation types that was constructed by MOTD, determines the machining method models for each $o \in OT_f$ and for feature f. Finally, MOMD constructs the machining method model $MOMM_i$ for the part type and posts it on BB. The pseudocode descriptions of agents SD, MOTD, and MOMD are shown in Figure 5.17, 5.18, and 5.19 (activity diagram is not included). The behavior of SD is the same as in Scenario 2a, but is repeated here for convenience.

Scenario 5 provides an example of how the efficiency of process plan construction can vary depending on the encapsulation of agent functions and the composition of the multiagent team. The work that was previously done by a single agent is now performed by two agent that can operate concurrently most of the time, while the complexity of communication does not appear to have increased significantly (for instance, the BB postings are the same as in Scenario 2a). Such division of work could be carried further by either having multiple identical agents sharing the load, or by specializing agent roles further to reduce the technological complexity of individual agents. Realistic examples in both directions can be described in CAPP World. The comparative performance of alternative multiagent systems (MAS) architectures constructed in this manner could then be studied through experiments using a CAPP World implementation.

```
agent Set-up Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        for $i \in \{1, \ldots, n\}$ do
            get BB.series-size(i) from $\mathcal{A}$;
        get BB.set-up-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make set-up-design-criteria;
    }
    foreach $i \in \{1 \ldots n\}$ do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.location-direction(f);
            get BB.prioritization-info(f);
        }
    }
    make ptps = part-type-priority-sequence;
    post BB.part-type-priority-sequence;
    foreach $i \in ptps$ do {
        when $MOTM_i$ posted
            for $f \in F_i$ do {
                get BB.operation-class(f);
                get BB.tool-access-directions(f);
            }
        for $k \in \{1, \ldots, 4\}$ do
            make side(i,k);
    }
    foreach $i \in ptps$ do {
        when $MOMM_i$ posted
            for $f \in F_i$ do
                get BB.feature-machining-summary(f);
        for $k \in \{1, \ldots, 4\}$ do
            make side-machining-summary(i,k);
    }
    make set-up-types;
    make set-up-plan;
    make set-up-programs;
    make process-plan;
    post BB.process-plan;
}
```

Figure 5.17: Scenario 5: The Set-up Designer (SD).

```
agent Machining Operation Type Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        get BB.operation-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make operation-design-criteria;
    }
    when part-type-priority-sequence posted
        get ptps = BB.part-type-priority-sequence;
    foreach $i \in$ ptps do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do {
            get BB.feature-description(f);
            make MOTM(f);
        }
        make $MOTM_i$;
        post BB.$MOTM_i$;
    }
}
```

Figure 5.18: Scenario 5: The Machining Operation Type Designer (MOTD).

```
agent Machining Operation Metod Designer {
    when $PPT$ posted {
        get $n$ =BB.assortment-size from $\mathcal{A}$;
        get BB.operation-constraints from $\mathcal{C}$;
        get BB.team-objectives from $\mathcal{R}$;
        make operation-design-criteria;
    }
    when part-type-priority-sequence posted
        get $ptps$ = BB.part-type-priority-sequence;
    foreach $i \in ptps$ do {
        when $MFM_i$ posted
            get $F_i$ =BB.feature-id-set($MFM_i$);
        for $f \in F_i$ do
            get BB.feature-description($f$);
        when $MOTM_i$ posted
            for $f \in F_i$ do {
                get $OT_f$ =BB.operation-type-id-set(MOTM($f$));
                for $o \in OT_f$ do {
                    get BB.opt-description($o$);
                    make MOMM($o$);
                }
                make MOMM($f$);
            }
        make $MOMM_i$;
        post BB.$MOMM_i$;
    }
}
```

Figure 5.19: Scenario 5: The Machining Operation Method Designer (MOMD).

5.8 Scenario 6: Varying Communication Mechanisms

In previous scenarios, the agents communicate exclusively through the blackboard, using the mechanisms of posting, publishing, notifications, and retrieval. This approach to communication has several advantages: agents do not receive unsolicited information; asynchronous interaction between activities is minimized as they are not engaged in dialogues with each other; and the communication bandwidth is used rationally as the agents retrieve only the required information. However, certain types of multiagent coordination arising in CAPP scenarios include negotiation, for which the blackboard mechanisms are not well suited. In this scenario we introduce direct communication through messages using a suitable multiagent communication language (such as Agent Communication Languages (ACL) specified by the Foundation for Intelligent Physical Agents (FIPA) (Fipa, 1997)). This allows agents to proactively initiate dialogs with team members asking questions and making suggestions. The expectation is that direct interaction between team members can reduce the effects of decision myopia through providing feedback and limited backtracking within same iteration of process plan construction. If this expectation is justified, satisfactory process plan could be reached in fewer iterations. In addition an iteration based on a seriously flawed strategy could be aborted rather then run into completion.

A simple example illustrating the use of a direct message communication in the core CAPP World can be based on Scenario 2a where the direct messaging may be established between Set-up Designer (SD) and Machining Operation Designer (MOD) during the formation of setup types. In order to make the set-up types that will satisfy the design criteria such as the number of tool instances per set-up, tool utilization per set-up, and the balancing machining time among different set-up types, SD waits for the posting of the $MOMM_i$ to make the side machining summary. After making the side machining summary for every part type from the assortment, SD makes the set-up types. At this point in the modified scenario, SD makes an assessment of the set-up types and may concludes that the tool utilization in some machining operations has been poor. SD immediately sends to MOD the direct message with the specific request regarding the utilization of tools in the particular *MOMM*s. After receiving the message, MOD confirms to SD the reception of the message and deliberates about the possible solutions to the SD's request. It then sends the response informing SD of either its new postings or of its failure to fulfil the request. If the request has been fulfilled (posting of new *MOMM*), SD reassesses the set-ups and either confirms the set-up types formation and proceeds to the next step or repeats the communication with MOD. In the case of failure, SD proceeds to the next step.

This scenario may be used for comparative studies of the influence that communication mechanisms have on the quality of the initial version of the generated process plan.

Chapter 6

Analysis and Evaluation

The main objective of this Thesis has been to define a microworld model for computer-aided process planning (CAPP) that is simple and manageable, representative for real CAPP, and suitable for multiagent systems (MAS) studies. In this chapter we analyze and evaluate the proposed CAPP World model with respect to these objectives.

The CAPP World presented in Chapter 4 consists of abstractions from the real CAPP systems. Those abstractions have been carefully chosen to maximally simplify and yet not trivialize the essential aspects of CAPP that must be preserved in a framework intended for domain-specific comparative studies of multiagent solutions and system architectures. Another criterion motivating specific choices has been to keep the microworld extensible in many directions. In order to assess to what extent these objectives have been achieved, we examine the design decisions of CAPP World with respect to the balance between simplicity and domain relevance.

The product class is represented by cube-shaped stock and uniform cylindrical features with the simplifying geometric assumption of all axes being in the same central plane. This greatly reduces both combinatorial and technological complexity, and yet preserves the notions of feature precedence, location direction, and tool access direction that all play major roles in process planning for real CAPP. It also naturally supports two major operation classes—milling and drilling. Most of our examples focus on drilling operation class, in which a simple feature is machined with two to three different operations. Milling operations can be left out or included to a varying degree to achieve the desired level of technological complexity. The sizes of assortments and series are not limited, allowing one to vary combinatorial complexity as desired.

The manufacturing cell provides pallets that can hold one, two, or four workpieces in a set-up. For an assortment with a single part type, the formation of set-ups requires moderately complex reasoning, with the difficulty increasing if multiple criteria are used. The criteria may include minimizing the total number of set-up instances, improving the economy of tool usage, and equalizing set-up processing times. It should be noted that these criteria relate set-up formation with different aspects of CAPP and different types of complexity. The first one is largely combinatorial, while the other two are technologically complex and logically interrelated. The combinatorial and logical complexity rise dramatically as the number of different part types increases. Set-up formation thus provides a wide range of issues and complexity levels that various domain-specific multiagent scenarios might require.

Other simplifications in the manufacturing cell include the single machine type (HMC) and the related assumption that each set-up instance is completely processed on a single machine. These assumptions can be relaxed without contradicting any other aspects of CAPP World design.

The actions in CAPP World are representative of real CAPP. They can be interpreted at different levels of complexity, ranging from relatively straightforward algorithms to agent goals motivating artificial intelligence (AI) research. In a typical multiagent scenario in CAPP World, most actions would have simple interpretations and straightforward implementations, with a small set selected for in-depth study being enhanced with additional elements that are not a part of the core CAPP World model.

In quantitative simulation studies, the approach in which some system components are highly simplified gives rise to the question of how one represents the performance of such components in the context of overall system performance. To this end one would need performance estimates for different actions when they are at a comparable level of complexity as the actions singled out for in-depth study, so that one could accurately assess the impact of alternative solutions in the selected area upon the performance of the CAPP system as a whole. This aspect of CAPP World has not yet been sufficiently explored.

Chapter 5 introduces a core multiagent system setting that complements the core CAPP World to produce a unified framework for multiagent CAPP studies. This setting includes a basic team composition consistent with an existing role classification for CAPP, and a basic communication and coordination model based on a well defined blackboard concept. Both the team structure and the communication and coordination model are essentially simple but extensible in many directions.

The scenarios provide a sampling of alternative multiagent solutions to relevant CAPP issues. The selection was partly guided by the areas identified in the literature as requiring further research. The scenarios address topics in the following areas: agent encapsulation; cooperation and coordination; cooperative iterative improvements of process plans; improving the efficiency of process planning through caching of design solutions; varying team compositions; and communication mechanisms. The examples span a fairly wide range of topics, including those recognized as critical to practical success and future deployment of multiagent systems for CAPP in industrial organizations.

The above analysis suggests that the proposed CAPP World model captures the relevant concepts from both CAPP knowledge domain and multiagent software technology. It also suggests that at the current level of exploration, the proposed microworld model provides a suitable framework for multiagent CAPP studies, and that it has achieved its objectives of simplicity and relevance. These observation remain to be further confirmed through studies involving prototype development and experimentation.

Chapter 7

Conclusions and Future Work

This Thesis proposes and investigates a novel framework for the study of multiagent solutions for computer-aided process planning (CAPP) in manufacturing systems. Preliminary research has included a review of literature in several areas: general planning, CAPP, multiagent systems (MAS) and, in particular, multiagent systems for CAPP. The main objective of that stage of work was to find out why, despite substantial advances in experimental multiagent systems for CAPP, there was much less visible progress towards deployment of practical systems in the manufacturing industry. The outcome was the observation that there was no simple unified framework in which the properties of multiagent solutions for CAPP could be investigated and compared. Another observation from the study of general planning and MAS literature was that fundamentals of a complex problem have often been successfully studied in a highly simplified abstract model, called microworld, that captured the essential characteristics of the problem. These observations have led to the main line of research in this Thesis, namely to the proposal and investigation of a domain-specific microworld for CAPP as a framework for comparative study of alternative multiagent solutions for building agent-based CAPP systems.

The first step towards a microworld for CAPP was to formulate the design objectives. The model had to be simple and manageable, integral in the sense of including the main aspects of CAPP, representative of real-world CAPP, and suitable for the investigation of multiagent design solutions relevant to CAPP systems. The next step was an analysis of complexity factors in CAPP, leading to decisions on what to include in the model and what to leave out. The inferred design principles then led to a concrete design of a CAPP microworld, called the CAPP World. The CAPP World is characterized by a product class, a model of a manufacturing cell, and appropriate adaptation and simplification of CAPP modeling concepts from the literature. These abstractions led to a collection of specific actions that jointly construct a process plan in CAPP World.

CAPP World was then submitted to a test of how well it can support domain-specific multiagent scenarios addressing a number of topics relevant to the development of MAS for CAPP. Scenarios of multiagent process planning have been developed that vary certain aspects of agent encapsulation, cooperation and coordination among team members, cooperative iterative improvement of process plan, improve the efficiency of process planning through caching of design solutions, team composition, and communication mechanisms. The constructed scenarios give rise to numerous variations that open additional relevant domain-specific questions. The analysis suggests that the established framework is indeed sufficiently simple and manageable, representative of relevant CAPP issues, and suitable for representing and investigating multiagent solutions, to allow one to conclude that CAPP World has met its design objectives.

In order to use the conceptual basis provided by CAPP World for building a practical platform for experimental work, two areas stand out among the several research topics that need to be addressed. The first area concerns performance metrics and real-world estimates for CAPP actions, so that quantitative results can be related to performance expectations in real CAPP systems. The second concerns the modeling of expert reasoning and teamwork in order to clearly represent how individual decision criteria can be synthesized from specialist knowledge and team-level planning requirements.

Appendix A

Notation

The pseudocode language used in our research is illustrated in Section A.1, and activity diagram notation is listed in Section A.2.

A.1 The Pseudocode Language

Agents operating in a computer-aided process planning (CAPP) system construct process plans on an active component called Blackboard (BB). BB is the agents' shared environment, which they *perceive* using ***get*** commands and *act upon* using ***post*** commands. In addition, an agent can subscribe to specific components of CAPP models and be notified by BB of their postings. Notifications are software signals that can be intercepted and processed within the agent in a variety of ways. In particular, in an agent that is internally multi-threaded, a notification can activate the thread scheduler and lead to a context switch when appropriate. An agent internally constructs CAPP model components using ***make*** commands, and then posts them if they need to be shared with others. Agents in a process planning team can communicate with each other through

BB, and do not strictly require other communication mechanisms in order to cooperate.

Multi-agent solutions of CAPP problems can employ additional mechanisms for interaction of agents with BB and each other. Such mechanisms can be selectively added as necessary in order to explore and compare the effectiveness of different approaches. An additional perception function could be *browsing*, allowing agents to have a wider view of the environment beyond a predefined collection of get functions. There could also be direct communication between agents using messages in an agent communication language (ACL).

Interaction between agents and BB is based on a common ontology of CAPP models involved in the construction of process plans. This allows BB to understand what an agent is requesting to extract through a ***get*** command, and what an agent is requesting to place on BB through a ***post*** command. In our pseudo-code for agent description, the commands have the syntax

get [var =] BB.<*perception function*>

post BB.<*model component*>

get retrieves from BB the information extracted by the specified function and places the information into predefined structures in the agent's belief space. For purely syntactic convenience, the returned information can also be optionally assigned to a program identifier in order to simplify the code. ***post*** transfers a CAPP model component from the agent that constructed it to BB and incorporates it into the previously developed model structures on BB. The effects of these commands are information transfers; they do not involve any design decisions.

The construction of CAPP model components is specified by commands of the form

make *<model component>*

Successful execution of a ***make*** command fulfills a subgoal in the overall development of a process plan. This may involve other, lower-level subgoals, i.e., the realization of a ***make*** command may invoke other ***make*** commands, as well as ***get*** and ***post*** if the realization of subgoals involves interaction with other agents. The top goal can be specified as

make *approved-process-plan*

indicating the construction of a process plan that fully realizes the process planning task $\mathcal{T} = (\mathcal{A}, \mathcal{C}, \mathcal{R})$, including the performance requirements $\mathcal{R}$. A first-level subgoal

make *process-plan*

aims at the construction of a process plan that satisfies the technical specifications and constraints of $\mathcal{A}$ and $\mathcal{C}$, but does not necessarily reach the performance objectives in $\mathcal{R}$. While the construction of *process plan* is primarily a problem in constraint satisfaction category, the construction of an *approved process plan* is largely an optimization problem.

The pseudo-code language includes several constructs that help specify coordination of activities within and between agents. The loop construct

foreach *<iterator>* **do** *<command>*

is normally meant to indicate execution by concurrent threads in an agent that is internally multi-threaded. Furthermore, the iterator can specify priorities that should be observed by the thread scheduler. If the agent is single-threaded, the order of execution of loop iterations is still controlled by the priority in the iterator construct. This mechanism helps implement interactions in which an agent observes the priorities set by another agent.

The synchronization of agent's activities with Blackboard postings is specified by the constructs

when <*model component*> **posted** <*command*>

if <*model component*> **posted** <*command*> [**else** <*command*>]

If the specified model component has not been posted, the first construct blocks and waits, while the second skips the command and proceeds or takes the **else** branch if specified.

Other language constructs will be introduced as necessary.

A.2 Activity Diagrams

The research uses Unified Modeling Language (UML) to illustrate cooperation among elements of the microworld model (Booch et al., 2005). The scenarios presented are demonstrated with pseudocode language (please see Section A.1) and UML activity diagrams. Dumas and ter Hofstede (2001) discussed UML activity diagrams as a workflow specification language, and we apply that concept in our study.

The UML activity diagram elements are constrained to the pseudocode language defined in Section A.1. The key elements of UML activity diagram in CAPP World are illustrated with pseudocode language as following:

1. As to UML activity diagram, the loop statement:

   foreach $i \in \{1 \ldots n\}$ **do** C;

   is illustrated as Figure A.1.

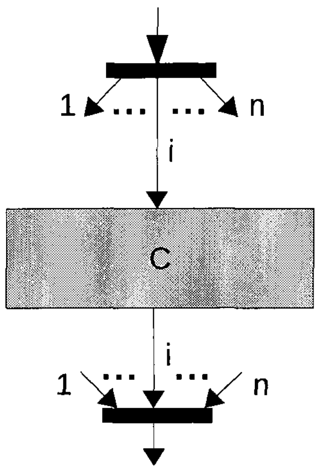


Figure A.1: **foreach** structure.

2. If "priority" notation is attached to a loop element, it means that the set $\{1 \ldots n\}$ has been ordered in terms of some priority weight. Figure A.2 shows such a modification of **foreach** structure:

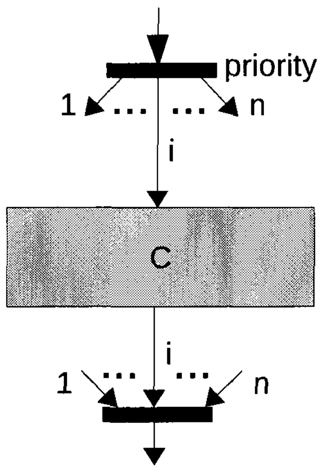


Figure A.2: Priority **foreach** structure.

3. The synchronization of agent's activities with Blackboard postings is specified by the constructs

when <*model component*> **posted** <*command*>

As to UML activity diagram, the statement:

when M **posted** C;

is illustrated as Figure A.3.

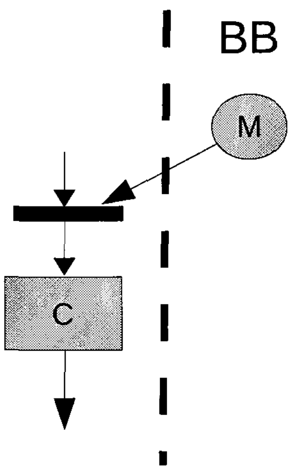


Figure A.3: **when** structure.

4. As to UML activity diagram, the statement:

   if M **posted** C;

   is illustrated as Figure A.4.

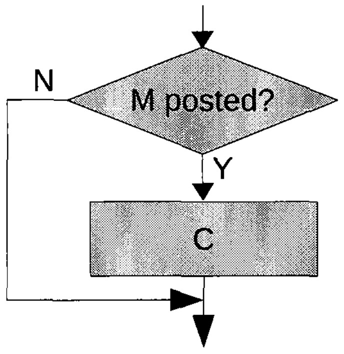


Figure A.4: **if** structure.

Table A.1 summaries those key elements of UML activity diagram in CAPP World. Other UML activity diagram elements will be introduced as necessary.

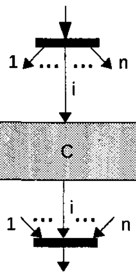
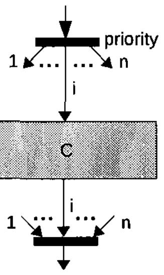
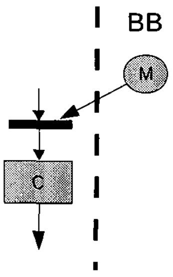
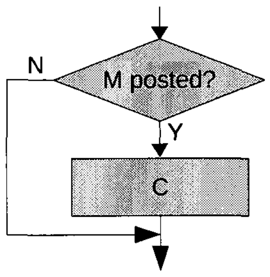
UML Activity Diagram Element	**Pseudocode Language**
	**foreach** <*iterator*> **do** <*command*>
	**foreach** <*iterator ordered by priority*> **do** <*command*>
	**when** <*model component*> **posted** <*command*>
	**if** <*model component*> **posted**

Table A.1: The elements of UML activity diagram in CAPP World.

Bibliography

F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1):123–191, 2000.

G. Booch, J. Rumbaugh, and I. Jacobson. *Unified modeling language user guide.* Addison-Wesley Professional, 2005.

U. Bose. A cooperative problem solving framework for computer-aided process planning. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 32, pages 289–289, 1999.

M. Bratman. *Intention, Plans, and Practical Reason.* Cambridge, Mass., Harvard University Press, 1987.

T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

D. Chapman. Planning for conjunctive goals. Master's thesis, Massachusetts Institute of Technology, 1985.

D.D. Corkill. Blackboard systems. *AI expert*, 6(9):40–47, 1991.

M. Dumas and A.H.M. ter Hofstede. UML activity diagrams as a workflow specification language. In *Proceedings of the 4th International Conference on The Unified Modeling*

Language, Modeling Languages, Concepts, and Tools, pages 76–90. Springer-Verlag London, UK, 2001.

M.D. Ernst, T.D. Millstein, and D.S. Weld. Automatic SAT-compilation of planning problems. In *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

R. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3/4):189–208, 1971.

F. Fipa. specification part 2: Agent communication language. *FIPA http://www.fipa.org/spec/FIPA97.html*, 1997.

M.P. Georgeff and A.L. Lansky. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682. Seattle, WA, 1987.

M.L. Ginsberg and D.F. Geddis. Is there any need for domain-dependent control information? In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 452–457, 1991.

G. Halevi. *Process and operation planning.* Kluwer Academic Pub, 2003.

J.A. Hendler, A. Tate, and M. Drummond. AI planning: Systems and techniques. *AI magazine*, 11(2):61–77, 1990.

H. Kautz and B. Selman. The role of domain-specific knowledge in the planning as satisfiability framework. In *Proceedings of the Forth International Conference on Artificial Intelligence Planning Systems*, pages 181–189. AAAI Press, 1998.

M. Mäntylä, D. Nau, and J. Shah. Challenges in feature-based manufacturing research. *Communications of the ACM*, 39(2):77–85, 1996.

S. Minton. Is There Any Need for Domain-Dependent Control Information?: A Reply. In *Proceedings of the National Conference on Artificial Intelligence*, pages 855–862, 1996.

L. Monostori, J. Váncza, and S.R.T. Kumara. Agent-based systems for manufacturing. *CIRP Annals-Manufacturing Technology*, 55(2):697–720, 2006.

H.P. Nii. *Blackboard systems.* Knowledge Systems Laboratory, Depts. of Medical and Computer Science, Stanford University, 1986a.

H.P. Nii. The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. *The AI Magazine*, 7(2):38–53, 1986b.

J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *proceedings of the third international conference on knowledge representation and reasoning*, pages 103–114, 1992.

A. Pokahr, L. Braubach, and W. Lamersdorf. JADEX: a BDI Reasoning Engine. *MULTIAGENT SYSTEMS ARTIFICIAL SOCIETIES AND SIMULATED ORGANIZATIONS*, 15:149, 2005.

D. Polajnar and J. Polajnar. A Formal Model of Manufacturing Process Plan for Chip-removal Metal-cutting Technologies. unpublished internal report, July, 2009.

D. Polajnar, J. Polajnar, and L. Lukić. Metamodel abstractions of agent roles in cooperative process planning. In *Proceedings of the 2008 IEEE SMC International Conference on Distributed Human-Machine Systems*, pages 77–82, Athens, Greece, 2008a.

D. Polajnar, J. Polajnar, L. Lukić, and M. Djapić. Complexity challenges in CAPP systems and promises of multi-agent technology. In *Proc. Sixth International Triennial Conference on Heavy Machinery*, Kraljevo, Serbia, June 2008b.

M.E. Pollack. The uses of plans. *Artificial Intelligence*, 57(1):43–68, 1992.

M. Reddy and GMP O'Hare. The blackboard model: a survey of its application. *Artificial Intelligence Review*, 5(3):169–186, 1991.

A.A.G. Requicha. Geometric reasoning for intelligent manufacturing. *Communications of the ACM*, 39(2):71, 1996.

L.P. Rieber. Computer-based microworlds: A bridge between constructivism and direct instruction. *Educational Technology Research and Development*, 40(1):93–106, 1992.

S. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Prentice Hall, 2003.

P. Scallan. *Process planning: the design/manufacture interface.* Butterworth-Heinemann, 2003.

W. Shen, L. Wang, and Q. Hao. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(4):563–577, 2006.

Y. Shoham and K. Leyton-Brown. *Multiagent systems: algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2009.

J. Slaney and S. Thiebaux. Linear time near-optimal planning in the blocks world. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1208–1214, 1996.

L. Wang, W. Shen, and Q. Hao. An overview of distributed process planning and its integration with scheduling. *International Journal of Computer Applications in Technology*, 26(1):3–14, 2006.

D.S. Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27–61, 1994.

M. Wooldridge. *An Introduction to MultiAgent Systems.* West Sussex, England: John Wiley and Sons Ltd, second edition, 2009.

M. Wooldridge and N.R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge engineering review*, 10(2):115–152, 1995.

F. Zambonelli, N.R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on software Engineering and Methodology*, 12 (3):317–370, 2003.

W.J. Zhang and S.Q. Xie. Agent Technology for Collaborative Process Planning: a Review. *The International Journal of Advanced Manufacturing Technology*, 32(3): 315–325, 2007.

F.L. Zhao, S.K. Tso, and P.S.Y. Wu. A cooperative agent modelling approach for process planning. *Computers in Industry*, 41(1):83–97, 2000.