Re-use of Interaction Protocols for Agent-based Control Applications

Stefan Bussmann[1], Nicholas R. Jennings[2], and Michael Wooldridge[3]

[1]DaimlerChrysler AG, Research Information and Communication
Alt-Moabit 96A, 10559 Berlin, Germany.
Stefan.Bussmann@daimlerchrysler.com

[2]Dept. of Electronics and Computer Science, University of Southampton
Southampton SO17 1BJ, United Kingdom.
nrj@ecs.soton.ac.uk

[3]Dept. of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom.
M.J.Wooldridge@csc.liv.ac.uk

Abstract. This paper presents a design method for re-using existing interaction protocols in agent-based control applications. In particular, this paper presents a general set of criteria for classifying interaction situations and matching them with existing interaction protocols that are able to resolve the interaction situations. This classification scheme is based solely on criteria derived from the specification of an interaction situation and thus enables a designer to select a suitable interaction protocol for these interaction problems without going through all the interaction protocols available. This design method completes the DACS methodology for agent-oriented analysis and design of control systems.

1 Introduction

The increasing industrial exploitation of agent technology in recent years has highlighted the importance of having agent-oriented software engineering frameworks. Put simply, they are necessary if agent technology is to be widely adopted. To provide such a framework, several agent-oriented methodologies and software engineering techniques have been developed (see e.g. [3,19]). To date, however, most agent-oriented design methodologies proposed have focused on developing an agent-based system from scratch. The methodologies either ignore the large body of agent-oriented techniques already available or leave it to the designer to identify and incorporate those techniques that may be useful in developing the envisioned agent-based system. Both of these situations, however, are undesirable. As with other areas of software [5,15], re-use could significantly improve matters.

To this end, this paper presents a design method for re-using existing interaction protocols. This design method addresses the first and most crucial step in re-use, namely the identification of those interaction protocols that could possibly be used in a design. To perform this identification step, the designer must have a mechanism that enables him to identify a suitable interaction protocol by only specifying his

interaction problem. In particular this needs to be achieved without going explicitly through all the existing interaction techniques and deciding for each one whether it is useful or not. Against this background, this paper presents a classification scheme which is based on criteria solely taken from the specification of an interaction problem and which, as a result, pinpoints to those interaction protocols that could possibly be used in the design. This work is couched in terms of the DACS (design of agent-based control systems) methodology we are developing for analysing and designing agent-based control systems [2].

The remainder of the paper is organised as follows. Section 2 recounts the basic concepts used by the DACS design method. Section 3 presents the main contribution of this paper – the design method for selecting interaction protocols. Section 4 discusses related work. Finally, the last section concludes with an evaluation of the method presented.

2 Overview of DACS Design Methodology

The goal of DACS is to enable an engineer with only limited training in agent technology and no prior experience in agent applications to design an agent-based control system. The engineer is given a description of the control problem to be solved, and then runs through the following three steps in order to design the agent-based system.

1. *Analysis of decision making* – The control decisions that are necessary to operate the target process are identified and analysed.
2. *Identification of agents* – The necessary agents of the control system, their decision responsibilities, and their interaction requirements are identified.
3. *Selection of interaction protocols* – A suitable interaction protocol is chosen for each situation in which the agents need to interact.

The first two steps have already been described in [2]. The third step is the contribution of this paper. The rest of this section describes those concepts developed in the previous work that are necessary to understand the third step.

The method for selecting the interaction protocols builds upon two concepts used to analyse the necessary decision making in step 1: *decision tasks* and *decision dependencies*. A **decision task** specifies a situation at the controlled process in which the controller must make a decision about which action to perform in this situation. A decision task is defined by a *trigger* indicating that the situation has occurred; a *decision space* listing the possible alternatives the controller has in this situation; and a set of *(local) constraints and preferences* on the decision space determining which actions are eligible and which are preferred.

Since control decisions can have far-reaching effects, the control decisions may be dependent on each other for finding the best control actions that create an optimal system performance. These **dependencies** are identified and characterised by specifying *non-local constraints and preferences*, i.e., constraints and preferences that involve several decision tasks. Whenever a dependency exists between decision tasks belonging to different agents, these agents need to interact in order to determine the decision alternative that not only satisfies the local, but also the non-local constraints

and preferences. To select an existing interaction protocol that is able to perform this interaction is the goal of the design method presented in this paper.

3 Selecting Interaction Protocols

To re-use existing interaction protocols, there must be a design method that enables the designer to select a suitable protocol given the description of a decision dependency between decision tasks. Such a design method must provide a set of criteria such that the interaction protocol which matches a dependency best – according to these criteria – is also the best interaction protocol to resolve the dependency. Given such a set of criteria, the designer only needs to classify a dependency according to these criteria and then search through a library of existing interaction techniques to find the interaction protocol that matches the classification best (see figure 1). In case, this library is computer-based, the search process may even be done automatically.

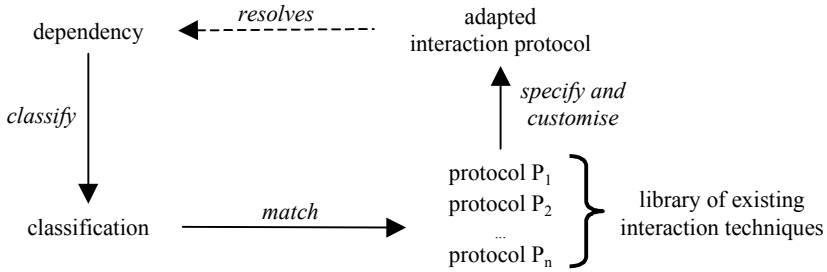

Fig. 1. The process for selecting interaction protocols.

The proposed process for selecting interaction protocols is a *heuristic classification* [4] because the selection mechanism is based on an abstract description of the interaction situation and the protocols. Such an abstraction is necessary if there is no direct matching between problem and solution (see [4] for a discussion). This direct link does not exist because a dependency may be solved by several interaction protocols.

To select an interaction protocol for a given dependency, the designer must consequently perform three steps. The first one is to classify the dependency according to a pre-defined set of criteria (called the ***classification scheme*** in what follows). The second step of the selection process is to match the classification of the dependency against a library of existing interaction techniques. A ***matching procedure*** specifies how the matching is performed and how, based on the results of this matching, the interaction protocol best suited to resolve the dependency is identified. To make such a matching possible, the existing interaction protocols must be classified according to the same criteria as the dependency. This process – which needs to be done only once for each interaction protocol – will be called ***protocol characterisation*** in the following. Once a suitable interaction protocol has been identified, the last step of the selection process is to specify it in terms of the

application and, if necessary, to adapt it to the specific requirements of the dependency situation. This final step will be referred to as the ***protocol customisation***.

The following subsections describe each aspect of the selection process in detail. The first subsection develops the classification scheme for dependencies. Subsection 3.2 shows how existing interaction protocols must be characterised in order to match the classification scheme and gives two examples of such characterisations. Subsection 3.3 then presents the procedure for matching dependencies to existing protocols, and explains to what extent a chosen protocol can be customised to fit an actual dependency situation in a given application.

3.1 Classification Scheme

The classification scheme is the core mechanism for re-using interaction protocols. To enable efficient re-use, this scheme should classify dependencies such that the interaction protocol which matches the classification of a dependency best is also the best interaction protocol to resolve the dependency. The classification scheme must consequently consist of classification criteria that put dependencies into different classes if they require different (kinds of) interaction protocols. To identify such a set of criteria, it is necessary to look at the requirements a dependency may impose on the interaction process and collect those aspects which differentiate dependencies most with respect to the required interaction process. This is the objective of this subsection.

A dependency consists of a set of decision tasks and a set of non-local constraints and preferences these decision tasks must fulfil (see section 2). Each decision task specifies a set of possible ***start situations*** in which the decision problem arises; and the decision tasks in combination with the non-local constraints and preferences specify what ***goal state*** must be achieved in the end. Any interaction protocol supposed to resolve the dependency must be able to reach the goal state from any possible start situation. Start situations and goal state of a dependency thus delineate the functionality of the required interaction protocol. It must be applicable to any start situation and must be able to achieve all aspects of the goal state. Both, start situations and goal state, are therefore analysed below in order to identify classification criteria distinguishing interaction protocols with respect to their applicability.

Start Situation. A start situation of a dependency is basically defined by two aspects: the decision tasks that share a dependency, and the constraints and preferences that restrict their decision making. For the selection of an interaction protocol, both aspects must be classified in application-independent terms. (Application-dependent criteria would limit the universality of the re-use mechanism and are also not appropriate because most interaction protocols are defined in general terms.)

Decision Tasks Involved in the Dependency. The first relevant criterion for the selection of a suitable interaction protocol is certainly the number of decision tasks that need to be co-ordinated. Is there, for instance, a small and fixed number of decision tasks that need to interact, or does the set of decision tasks change over time? Since dependent decision tasks only need to be co-ordinated if they belong to

different agents, the first relevant criterion for selecting interaction protocols is therefore the number of agents involved in the dependency.

Criterion #1: ***Number of agents involved***

How many agents are involved in the dependency right from the start? May other agents join later?

The possible answers to the above questions are classified according to the requirements they impose on the required interaction process.

fix The number of agents involved in the dependency is fixed at the beginning of the interaction.

changing The number of agents involved may change during the interaction, i.e., agents may join the interaction process after it has been started. Agents may join later, for example, because they have been introduced to the control system after the beginning of the interaction.

The second class – *changing* – imposes a stronger requirement on the interaction process than the first class. When the number of agents involved is fixed, the interaction protocol chosen must be able to deal with an arbitrary, but fix number of agents. In the special case that the number of agents involved is fixed and already known at design time, the designer may even choose an interaction protocol that is only able to deal with the number of agents indicated. Some interaction protocols, for instance, are only able to co-ordinate two agents. In case the number of agents is not fixed, but changing, the protocol must additionally be able to integrate new agents into the interaction process after it has been initiated.

Relation of Constraints and Preferences. The other important aspect of the start situation is how the decision tasks involved in the dependency are related to each other. Each agent has its local decision tasks, but is not able to execute them locally because of the non-local constraints and preferences that restrict the local decision making. As a consequence, the agents need to interact. The nature of the restrictions on the local decision making, however, have an influence on the kind of interaction required to deal with these restrictions. Agents that have completely opposing interests will have to interact more than agents that just want to avoid some damaging actions. The second relevant criterion for the selection of a suitable interaction protocol is therefore the relation of local and non-local constraints and preferences.

Criterion #2: ***Compatibility of constraints and preferences***

How compatible are the local and non-local constraints and preferences involved in a dependency?

The compatibility is classified according to the kinds of restrictions that create the dependency:

only constraints There are only constraints. These constraints – by definition – only rule out certain combinations of decision alternatives. Any combination of decision alternatives that is not ruled out is a solution resolving the dependency. Naturally, there may exist no solution satisfying all constraints.

compatible preferences There exists at least one (local or non-local) preference function on the outcome of the interaction (and possibly additional constraints). In case of more than one preference function, there are solutions that are to the mutual benefit of all agents, i.e., that satisfy all preference functions.

opposing preferences There are at least two agents that have preferences on the outcome of the interaction and these preferences are opposing, i.e., there is no combination of decision alternatives that maximises all local and non-local preference functions. (Constraints may be present or not.)

Another important aspect of the constraints and preferences linking the decision tasks is to what extent these constraints and preferences are global, i.e., encompass all decision tasks of a dependency. By definition, the non-local constraints and preferences involve at least two decision tasks. However, if there are more than two agents, the non-local constraints and preferences may involve all agents and thus be global, or only link subsets of the agents. This distinction is particularly relevant if there are many agents. In such a case, it may be far easier to co-ordinate small subsets of these agents than to make sure that all agents satisfy a global constraint or maximise a global preference function. Therefore, the start situation is also classified according to the existence of global constraints and preferences.

Criterion #3: ***Global constraints and preferences***

In case there are more than two agents, does there exist a global constraint or preference that involves all decision tasks?

The cases in which there are more than two agents and a global constraint or preference exists, are indicated by ***global***. All other cases are defined as ***non-local***.

Goal State. To resolve a dependency, the relevant agents need to choose an action for each decision task such that the local and non-local constraints and preferences are satisfied in the best manner possible. The goal state of a dependency is thus specified by a list of actions – one for each decision task. At least something about this goal state must be unknown at the start in order to represent a decision problem. Thus it will either be unclear which actions are to be taken by each agent or, if the decision spaces include the null action, which agents will be taking an action at all (otherwise the agents do not have a decision task). The interaction protocol to be selected will have to answer whichever question is unanswered at the beginning of the interaction. The first question – which action should be executed – however will be unanswered in most cases, and will therefore hardly distinguish interaction situations. The second question – which agent should commit to an action – on the other hand, may or may not be clear at the beginning. The second question is thus not common to all interaction situations and may consequently be used to distinguish dependencies with respect to the requirements they impose on the interaction protocol. This will be done below (see *role variability*).

The second important aspect of a goal state is how the actual decisions made relate to each other. Not necessarily every decision will equally depend on every other decision involved in the dependency. Consequently, at the end of the interaction not every agent will have to commit itself in front of everybody else to the decisions

made (even if they are all dependent on each other). Maybe some agents form a subgroup that is independent in their execution of the rest of the agents involved in the dependency. The number and size of the required joint commitments, however, is relevant to the selection of a suitable interaction protocol. Bilateral joint commitments are easier to achieve than a joint commitment encompassing all agents. The required joint commitments are therefore also analysed below (see *joint commitments*).

The criteria for classifying the joint commitments are presented first because any interaction situation requires joint commitments to be made.

Joint Commitments. In the context of this work, a set of commitments is called a **joint commitment** if the failure to fulfil one of the commitments jeopardises the success of the other commitments. That is, the set of commitments only makes sense if all commitments are fulfilled. If one agent de-commits, all other agents should de-commit, too.

Formally, joint commitments are represented by subsets of the agents involved in a dependency. If one agent of such a subset de-commits, all other agents in this subset should de-commit, too. The joint commitments required by a dependency may thus have quite diverse structures – namely any subset of the power set of the agents is theoretically a possible set of joint commitments. However, to make a comparison of joint commitments feasible and efficient, the classification of the required joint commitments is reduced to two criteria: the number of (independent) joint commitments, and the size of the commitments.

Criterion #4: ***Number of joint commitments***

Is the number of joint commitments required in the goal state already known at the beginning of the interaction, or must it be determined by the interaction protocol?

The possible answers to the above question are indicated as follows:

fix	The number of required joint commitments is known at the beginning of the interaction.
variable	The number of required joint commitments must be determined by the interaction protocol.

Criterion #5: ***Size of joint commitments***

How many agents are involved in a joint commitment? Do all joint commitments have the same size?

The possible answers to the above question are indicated as follows:

fix	All joint commitments have the same size.
differing	The joint commitments may have different size.
variable	The size of the joint commitments must be determined by the interaction protocol.

Role Variability. The goal state is described by a set of agent-action pairs, specifying which agent is executing which action. As discussed above, it may be unclear which of the agents available in the interaction situation will actually perform an action, and thus will be a member of one of the agent-action pairs. To capture this potential

uncertainty, the goal state will be characterised with the help of roles [10]. A role describes a specific behaviour without specifying which agent will actually perform this behaviour. In this view, the goal state consists of a set of roles, each specifying an action, and one task of the interaction protocol – apart from identifying these actions – is to assign these roles to agents. To classify this assignment problem for a given dependency, it is necessary to identify which roles are already assigned to agents and which must be assigned during the interaction process.

Criterion #6: *Role assignment*

Is an agent role already assigned to an agent, or must the role assignment be determined by the interaction protocol?

For each role, there are two possible answers: A role is either *fix* or *variable*. The classification of the agent roles can therefore be summarised by stating how many roles are variable (all others then must be fixed). Three cases are distinguished:

none	None of the agent roles are variable.
subset	A subset of the agent roles is variable.
all	All agent roles are variable.

The variability of a role is relevant to the selection of an interaction protocol because a variable role requires that the interaction protocol must not only choose an appropriate action, but must also find an agent to execute it. It is also relevant how many of the agent roles are variable because it is easier to assign some roles than all roles. Who will perform the role assignment if all roles (including the role of assigning the roles) is variable?

Summary. This section has identified six classification criteria that characterise decision dependencies with respect to the interaction process they require. These criteria define 216 possible classifications – namely the product of the possible classifications for each criterion. Due to the diverse aspects covered, these classifications already cover a wide range of different dependency situations. More criteria, however, can be defined and added to the classification scheme if necessary. How many, and in particular which criteria are necessary in order to optimally match interaction situations with interaction protocols ultimately depends on the type of dependencies encountered in an application and on the types of interaction protocols existing. Our experience, however, shows that the criteria presented here provide a sufficient basis for reducing the set of suitable interaction protocols to a small set (which then can be assessed manually).

3.2 Characterising Interaction Protocols

The re-use mechanism proposed in this paper requires that existing interaction protocols are characterised according to the same criteria as the interaction situations (see beginning of this section). Once such a characterisation of the existing interaction protocols is given (and it needs to be done only once for each protocol), the most suitable interaction protocol can be identified by matching the classification of the dependency against a library of existing interaction protocols.

The characterisation of an interaction protocol, though, is not simply a classification according to the scheme presented in the previous subsection. Instead of assigning it to a specific class of dependencies, an interaction protocol should be assigned to all those classes which it can efficiently solve. The task of the characterisation is therefore to analyse the interaction protocol with respect to the classes of dependencies it could possibly address.

As a first step towards a library of existing interaction protocols, a diverse set of protocols has already been characterised. This set includes protocols from consensus formation, bargaining, auction theory (in particular, one-sided and continuous double auctions), negotiation, distributed constraint satisfaction, coalition formation, and distributed planning. Due to space limitations, only two examples of protocol characterisations can be given in this paper. The protocols presented below were chosen because they are characterised quite differently.

The contract net protocol. The contract net protocol (CNP) is a simple, but efficient protocol for assigning tasks to individual nodes in a network [18]. It assumes that one node has a task that needs to be executed and that there are several nodes that are able to execute this task. The node with the task is called the *manager* and the other nodes are (potential) *contractors*. The manager initiates the protocol by announcing the task to the potential contractors, which answer with a bid. The manager compares the bids and chooses the best bid according to its preferences. The node which has sent the best bid then receives an award message and is said to have a contract with the manager about the execution of the task. The other nodes may or may not receive a reject message.

Criterion #1 – Number of agents involved: **fix**

The CNP involves several agents, namely one manager and at least two bidders. The number of bidders must be fixed at the beginning of the interaction because the manager announces the task to be contracted only once to exactly these bidders (of course, the protocol may be changed to accommodate a changing set of bidders).

Criterion #2 – Compatibility of preferences: **compatible**

The constraints and preferences of the different agents must be at least compatible. If the preference were opposing, it would not be possible to find a mutually acceptable compromise with the first bid.

Criterion #3 – Global constraints and preferences **non-local**

The CNP is not able to handle global preferences because each agent (i.e., manager and contractors) only take into account their local decision preferences.

Criterion #4 – Number of joint commitments: **1**
Criterion #5 – Size of joint commitments: **2**
Criterion #6 – Role assignment: **1/1**

There is only one joint commitment in the goal state, namely that of the manager and the contractor that wins the contract. Obviously, the size of this joint

commitment is two and it consists of only two roles. The first role, i.e., that of the manager, is fixed and the other role variable.

The relation of agent roles and joint commitments in the CNP is schematically exemplified in the following figure.

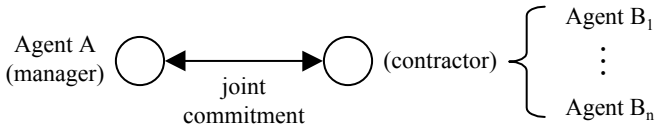

Fig. 2. The decision structure of the CNP.

Partial global planning. Partial global planning (PGP) was developed to co-ordinate distributed planners for sensory interpretation, each executing its own local plan for the interpretation of the distributed data [7]. To achieve the co-ordination of the distributed planners efficiently, the agents abstract from their plans and exchange these abstractions. Given the different local plan abstractions, each agent is then able to identify common goals to which the local goals of the agents contribute. Since these common goals may be only partially known to the agents, they are called *partial global goals*. Once a partial global goal has been identified, the local plans can be integrated into partial global plans. PGP in its original description provides two mechanisms to perform this integration: redundant tasks are avoided, and tasks are performed earlier if this facilitates the work of other agents. In contrast to many other interaction protocols, PGP is therefore an on-going mechanism for global co-ordination:

Criterion #1 – Number of agents involved: **changing**

Since the planning process is on-going and intertwined with the execution, agents may join the planning process at any time.

Criterion #2 – Compatibility of preferences: **compatible**

PGP is designed for co-operative agents. There is no mechanism in PGP to reconcile opposing interests.

Criterion #3 – Global constraints and preferences **global**

During the planning process, the agents construct (partial) global plans and try to optimise the overall system behaviour.

Criterion #4 – Number of joint commitments: **variable**
Criterion #5 – Size of joint commitments: **variable**
Criterion #6 – Role assignment: **all variable**

The number of joint commitments and their size depends on the global goals, i.e., the dependencies, that are identified. Since the local plans may be changed when integrated into the partial global plans, the roles of the agents may change also.

3.3 Matching and Customising Interaction Protocols

Given a library of existing interaction protocols characterised according to the classification scheme, the designer is now able to run through the following steps in order to select a suitable interaction protocol for a given dependency.

1. Collect all decision tasks involved in the dependency.
2. Identify all possible start situations in which this dependency may arise.
3. Perform the classification of the dependency.
4. Given a library of characterised interaction protocols, search for the interaction protocols that best match the classification of the dependency. An interaction protocol matches a dependency best if its classification has the most attributes in common with the classification of the dependency.
5. For each protocol identified, verify whether it is able to reach the goal state from all possible start situations. If this is not the case for a protocol, try to modify the protocol accordingly (see below).
6. Choose the interaction protocol that resolves the dependency best (after the customisation) and specify the (possibly adapted) interaction protocol (e.g., using the specification language presented in [1]).

If all six steps of the above method are successfully completed, the designer has found an interaction protocol that resolves the dependency and has thus solved the design task (concerning the interaction situation). The above method, however, may fail to identify a suitable interaction protocol for a dependency. This may have two reasons:

- It is not possible to resolve the dependency without resolving simultaneously other dependencies the decision tasks are involved in. In such a case, the above method has to be repeated with an enlarged scope. That is, in step one of the method all decision tasks involved in the set of (potentially) relevant dependencies are collected.
- It is possible to resolve the dependency, but there are no suitable interaction protocols in the library available to the designer. In this case, a new interaction protocol must be designed (or the identification of the control agents must be revised in order to arrive at a different set of dependencies).

Customising Interaction Protocols. For each interaction protocol that matches the dependency classification, it must be verified whether this protocol is able to reach the goal state from all possible start situations. An interaction protocol may fail to do so either because it is not applicable to one of the start situations, or because it does not reach the desired goal state. In the latter case, the designer must either redesign the protocol or choose a different protocol. In the former case, it may be possible – either at design or at run time – to transform the actual start situation into one to which the protocol can be applied. Here, two aspects are discussed.

First of all, the agents supposed to initiate the interaction protocol do not receive a trigger, or too many agents initiate the interaction protocol. In both cases, the interaction protocol must be preceded by a phase in which either the triggered agents inform the agents supposed to initiate the interaction protocol, or, in the second case, the agents clarify who should initiate the interaction process (e.g., through a voting process [16]).

Secondly, an agent may not have sufficient knowledge to perform its role in the decision making process. In such a case, the agents may have to gather (or compute) information before they can engage in the actual decision making protocol. As for decision making protocols, there are also a vast number of interaction protocols which are able to gather information in an agent-based system [11].

3.4 Examples

This section gives two examples for matching the classification of a dependency with a suitable interaction protocol. The example dependencies are taken from two real-world control applications at DaimlerChrysler – the first application is already in operation, the second is currently being prototyped.

Choosing a machine. For the first example, assume that a workpiece must choose a machine to perform the next set of operations. Further assume that an agent is associated with the workpiece and each machine. For choosing the next machine, there is consequently a dependency between the workpiece agent – which wants to choose a machine – and the machine agents – which must accept the workpiece for processing. Finally, assume that there is only one start situation, namely the workpiece agent is looking for a machine. The classification of this dependency is then as follows.

Criterion #1 – Number of agents involved: **fix**

There is a fix number of agents, namely the workpiece agent and all machine agents in the production system that could possibly process the workpiece.

Criterion #2 – Compatibility of preferences: **compatible**
Criterion #3 – Global constraints and preferences **non-local**

Constraints and preferences are assumed to be compatible because the workpiece agent wants to get processed and the machine agents want to offer processing (however, it may not be that simple in all control applications!). Furthermore, there are no global constraints or preferences; each agent is trying to optimise its performance.

Criterion #4 – Number of joint commitments: **1**
Criterion #5 – Size of joint commitments: **2**
Criterion #6 – Role assignment: **1/1**

The dependency is resolved if the workpiece has identified a machine that is most suitable for processing the workpiece and the machine has agreed to process it. Consequently, the agents are searching for a single joint commitment between two agents. The first role of the joint commitment, the workpiece agent, is obviously fixed, and the second role, that of the machine, is to be determined.

The above classification matches perfectly to the contract net protocol (see section 3.2), even though there exist other protocols, such as voting or auction protocols, that also match well with the above classification. A short analysis, however, shows that the CNP is sufficient to resolve the dependency. There is also no need to customise

the CNP. (Due to space limitations, the last step – specifying the interaction protocol is omitted.)

Meeting deadlines. For the second example, assume that each workpiece in a manufacturing system must meet a deadline for its delivery to the customer. Furthermore assume that each workpiece must run through several machines and that it uses the interaction protocol identified in the previous example to choose the next machine. Since the workpieces may compete for the machines when trying to meet their deadlines, there exists a dependency between all workpieces (and all their decision tasks to choose the next machine) in that the workpieces should resolve these conflicts such that the average tardiness, i.e., the average deadline violation, is minimised. This dependency is classified as follows:

Criterion #1 – Number of agents involved: **changing**

There is a changing set of agents since a new workpiece agent is created every time a new workpiece enters the manufacturing system.

Criterion #2 – Compatibility of preferences: **opposing**
Criterion #3 – Global constraints and preferences **global**

The workpiece agents may run into conflicts concerning the machine usage that cannot be resolved without one workpiece missing its deadline. The global preference, of course, is to minimise the average tardiness.

Criterion #4 – Number of joint commitments: **variable**
Criterion #5 – Size of joint commitments: **differing**
Criterion #6 – Role assignment: **some variable**

Joint commitments are constantly formed as workpiece agents enter the manufacturing system. In particular, each workpiece agent will engage in several joint commitments with machine agents. The joint commitments, though, may include more than two agents in case several workpiece agents resolve a resource conflict by agreeing on a certain order for using the conflict resource. The size of the commitments is therefore differing. Finally, the roles of the workpiece agents are all fixed, but the roles of the machines are not.

With respect to the existing interaction protocols characterised so far, the above classification matches best with the partial global planning approach which fully matches or subsumes the above classification (see section 3.2). Other interaction protocols, such as the continuous double auction or distributed constraint satisfaction, have less correspondence.

Conceptually, the PGP approach is also able to resolve the above problem of meeting deadlines. Several agents follow their plans to meet a certain deadline by allocating resources and may run into conflicts with other agents. These conflicts must be identified and resolved by the interaction protocol, just as PGP does so for distributed hypothesis formation. Since PGP was designed for distributed hypothesis formation, though, the interaction approach of PGP must be adapted to accommodate the peculiarities of the above problem (the same is true for GPGP as presented in [6]). Firstly, conflicts occur because of an overloaded resource. And secondly, conflicts

must be resolved by determining – possibly through negotiation – which workpiece has a higher priority. Strictly speaking, PGP and GPGP thus do not resolve the above dependency because the necessary changes go beyond protocol customisation as defined in the previous subsection. However, with PGP/GPGP a general framework has been identified that provides a basis for developing an adapted interaction protocol.

4 Related Work

The work on interaction-oriented programming has proposed analysis and design methods that use interactions as a basic concept for structuring an agent-based system (see e.g. [8,14]). These approaches put emphasis on the necessary interactions in an agent-based system and use concepts like team modelling or goal decomposition to identify the need for interaction. So far, however, this work has not addressed the aspect of identifying existing interaction protocols able to satisfy this need.

An increasing amount of work has been invested in the development of design patterns for agent-based systems (see e.g. [9,12] for the concepts). In this work, concepts of agent-based systems are specified in a general format in order to allow the re-use of these patterns. The work on design patterns is thus complementary to our work. While design patterns provide the re-usable interaction protocols, our design method explains how to choose the right interaction protocol for the design problem at hand.

Several researchers have developed taxonomies for classifying dependencies (see e.g. [13,17]). But despite their ground-breaking work, these classifications are not sufficient for re-using interaction protocols. Malone and Crowston [13], for instance, only provide a detailed taxonomy for tasks having resource conflicts. Their taxonomy does not cover state or preference conflicts (e.g., two workpieces which need to be assembled into one agreeing on the goal station). Malone and Crowston themselves do not claim to provide a complete taxonomy; for the re-use of interaction protocols, though, a complete taxonomy of interaction protocols is required (even if it is less detailed).

5 Conclusion

This paper has presented a method for re-using existing interaction protocols during the design of agent-based control applications. The main contribution of this work is a set of classification criteria that extracts the general requirements of an interaction situation on the interaction protocol to be used. The classification criteria are easily applied to an interaction situation because they were derived from the general specification of such situations. It is therefore relatively straightforward for a designer following the first two steps of the DACS methodology to perform the classification of each interaction situation. How this is done was shown with the help of two example dependencies from real-world control applications at DaimlerChrysler. Furthermore, the paper has shown – due to the space limitations with only two examples – that the classification scheme puts conceptually different interaction

protocols into different classes. The classification scheme thus enables a designer to select a suitable interaction protocol for a given interaction situation and thus to re-use existing interaction protocols he is not familiar with.

This claim has been validated in several real-world control applications – most of which have led to the implementation of a realistic simulation. After the first two applications the method has been revised considerably and the result has been presented here. To complete the evaluation, it is planned to test the complete DACS methodology with engineers designing control systems. Once these tests are successfully completed, the methodology can be released to development teams.

Nevertheless, it is not expected that the design method presented in this paper will remain unaltered after release. First of all, new interaction protocols will be developed in the future and, once characterised, should be added to the protocol library to enlarge the set of protocols that can be re-used. Secondly, new classification criteria may have to be added in the future if the newly developed interaction protocols fall into a single class of the existing classification scheme. Such an extension of the classification scheme could, for example, address further variations of the distributed constraint satisfaction technique. The work presented in this paper, though, has developed the concepts and the basic criteria for re-using existing interaction protocols.

References

1. B. Burmeister, A. Haddadi, K. Sundermeyer: Generic Configurable Cooperation Protocols for Multi-Agent Systems. In C. Castelfranchi, J.-P. Müller (eds.), *From Cognition to Action*, LNAI 957, pp. 157 – 171. Springer-Verlag, 1995.
2. S. Bussmann, N.R. Jennings, M.J. Wooldridge: On the Identification of Agents in the Design of Production Control Systems. In [3], pp. 141 – 162.
3. P. Ciancarini, M.J. Wooldridge (eds.), *Agent-Oriented Software Engineering*, LNCS 1957. Springer-Verlag, 2001.
4. W.J. Clancey: Heuristic Classification. In *Artificial Intelligence*, Vol. 27, pp. 289 – 350, 1985.
5. B. Coulange: *Software Reuse*. Springer-Verlag, 1998.
6. K.S. Decker, V.R. Lesser: Designing a Family of Coordination Algorithms. In *Proc. of the First Int. Conf. on Multi-Agent Systems*, pp. 73 – 80. San Francisco, USA, 1995.
7. E.H. Durfee: Planning in Distributed Artificial Intelligence. In G.M.P. O'Hare, N.R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, pp. 231 – 245. John Wiley & Sons, 1996.
8. M.N. Huhns: Interaction-Oriented Programming. In [3], pp. 29 – 44.
9. E.A. Kendall: Role Models: Patterns of Agent Analysis and Design. In *British Telecom Technical Journal*, 1999.
10. E.A. Kendall: Agent Software Engineering with Role Modelling. In [3], pp. 163 – 169.
11. M. Klusch: Information Agent Technology for the Internet: A Survey. In *Journal on Data and Knowledge Engineering*, Vol. 36, No. 3, 2001.
12. J. Lind: Patterns in Agent-Oriented Software Engineering. In this volume.
13. T.W. Malone, K. Crowston: The Interdisciplinary Study of Coordination. In *ACM Computing Surveys*, Vol. 26, No. 1, pp. 87 – 119, 1994.
14. S. Miles, M. Joy, M. Luck: Designing Agent-Oriented Systems by Analysing Agent Interactions. In [3], pp. 171 – 183.

15. H. Mili, F. Mili, A. Mili: Reusing Software: Issues and Research Directions. In *IEEE Trans. on Software Engineering*, Vol. 21, No. 6, pp. 528 – 561, 1995.
16. T.W. Sandholm: Distributed Rational Decision Making. In G. Weiss (ed.), *Multi-Agents Systems*, pp. 201 – 258. MIT Press, 1999.
17. J.S. Sichman, R. Conte, C. Castelfranchi, Y. Demazeau: A Social Reasoning Mechanism Based On Dependence Networks. In *Proc. of the 11th European Conf. on Artificial Intelligence*, pp. 188 – 192. John Wiley & Sons, 1994.
18. R.G. Smith: The contract net protocol: High-level communication and control in distributed problem solving. In *IEEE Transactions on Computers*, Vol. C-29, No. 12, pp. 1104 – 1113, 1980.
19. M.J. Wooldridge, G. Weiß, P. Ciancarini (eds.): *Agent-Oriented Software Engineering II*, LNCS 2222. Springer-Verlag, 2002.