14

Future Directions: Building a Decision Making Framework Using Agent Teams

Jeffrey Tweedale[1], Christos Sioutis[2], Gloria Phillips-Wren[3], Nikhil Ichalkaranje[4], Pierre Urlings[2], and Lakhmi C. Jain[1]

[1] School of Electrical and Information Engineering, Knowledge Based Intelligent Engineering Systems Centre, University of South Australia, Mawson Lakes, SA 5095, Australia
[2] Air Operations Division, Defence Science and Technology Organisation, Edinburgh SA 5111, Australia
[3] The Sellinger School of Business and Management, Loyola College in Maryland, Baltimore, MD 21210 USA
[4] School of Electrical and Information Engineering, University of South Australia, Adelaide, South Australia SA 5095, Australia

Summary. This chapter describes initial efforts and research directions in decision support systems that allow collaboration and cooperation between intelligent agents in a multi-agent system and humans. Description of previous research is included to show how developments in the agent software framework was implemented based on cognitive hybrid reasoning and learning models where decision support systems are used to support the human's roles. Cooperation is a type of relationship within structured teams when an agent is required to coordinate with, and explicitly trust, instructions and information received from controlling agents. Collaboration involves the creation of temporary relationships between different agents and/or humans that allow each member to achieve his own goals. Due to the inherent physical separation between humans and agents, the concept of collaboration has been identified as the means of realizing human-agent teams to assist with decision making. An example application and preliminary demonstration to show the current status is also presented. Future research needed to advance the field of intelligent decision support systems is identified.

14.1 Introduction

Decision Support Systems (DSSs) emerged in the early 1970s to assist and support humans in the decision making process. DSSs were initially generated by computer programmers in an attempt to capture the knowledge of subject matter experts in an information management system that could ideally be used to assist management in making decisions without the need for consultation or detailed analysis. The number of applications has expanded as

J. Tweedale et al.: *Future Directions: Building a Decision Making Framework Using Agent Teams*, Studies in Computational Intelligence (SCI) **97**, 387–408 (2008)
www.springerlink.com

computers have become ubiquitous and essential in professional and personal tasks. Recent advances in Artificial Intelligence (AI) have provided a new set of techniques and methods for DSSs that increase their scope and effectiveness. The chapters in this book attest to the intriguing possibilities of Intelligent Decision Support Systems (IDSSs) as combinations of DSSs and AI techniques to effectively support human decision making in complex environments. In this chapter we discuss some of the potential future developments in IDSSs.

One of the more promising areas of AI research for incorporation in IDSSs is intelligent software agents (or just agents). As indicated by Russell and Norvig (2003), an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. Agent-oriented development can be considered as the successor of object-oriented development when applied in AI problem domains. Agents embody a software development paradigm that attempts to merge some of the theories developed in AI research within computer science. Bratman's Beliefs, Desires, Intentions (BDI) reasoning model (Bratman 1999) has demonstrated the potential of becoming the method of choice for realizing truly autonomous agents. *Beliefs* represent the agent's understanding of the external world; *desires* represent the goals that it needs to achieve; and *intentions* are the courses of action that the agent has committed to follow in order to satisfy its desires (Rao and George, 1995).

When defining the intelligence of agents, researchers generally state the properties that a system of agents should exhibit. Firstly, *autonomy* means operating without the direct intervention of humans. Secondly, *social ability* means interacting with other agents. Thirdly, *reactivity* means perceiving their environment and responding to any changes that occur in it. Finally, *pro-activeness* means exhibiting goal-directed behavior (Wooldridge 2002). The social ability of agents provides the potential to create stand-alone or cooperative agents that communicate with other agents as required. Different techniques have been developed allowing agents to form teams, and agents can be dynamically assigned a particular role depending on the situation and their suitability. Recent advances in this field have focused on the formation of rather unique teams with human and machine members based on cognitive principles. One major advantage of such teams is an improved situation awareness capability for the human when dealing with unknown or hostile environments (Urlings, 2003).

This chapter focuses on the design of intelligent agent architectures. Agent teaming ability is illustrated with a simulation environment relevant for Airborne Mission Systems. Agent teaming has gained popularity in recent years and is categorised into the prominent domain of Multi-Agent System (MAS). It is believed that three important aspects, 'Communication, Coordination and Cooperation', play an important role in agent teaming. Multi-agent teaming takes inspiration from human organisational models of team operation, where leadership, communication, cooperation and collaboration skills empower the success of the team. In addition, future research directions and needs are identified.

14.2 Models of Decision Making

Information overload occurs when the amount of information available to the user for decision making is more than can be processed in a relevant time period. It is often associated with real-time decision making in which information changes rapidly, the quantity of information is large, and the relationships between the data items are difficult to discern. The Observe – Orient - Decide - Act (OODA) loop, also know as the four box method, shown in Fig. 14.1, is one approach used to aid humans in making decisions when overloaded with information. The cycle was originally labeled by Boyd as the OODA loop to assist pilots, as military decision-makers, to achieve knowledge superiority and avoid information overload in order to win the battle (Coram 2002). Boyd studied air-to-air engagements of the Korean War (1950–1953) in which US fighter pilots, despite flying F-86 Sabre aircraft with wider turn radii, had a consistent 10:1 victory ratio over MiG-15 aircraft that had much better manoeuvrability.

While conventional wisdom suggested that US pilots were successful because they were better trained, Boyd suspected it was due to much more. His hypothesis was that a US pilot would win almost every dogfight because he could complete loops of decision-making much faster than his adversary. Boyd constructed such a loop with the four distinct steps shown in Fig. 14.1 (Curts and Campbell, 2001):

Observe - US pilots could see their adversaries earlier and better because the cockpit design of their aircraft ensured better visibility.

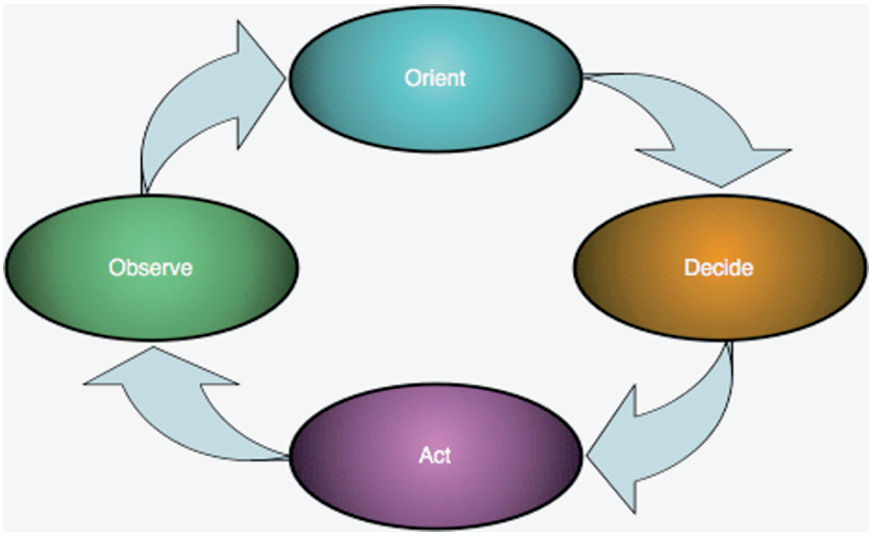

Fig. 14.1. Boyd's observe-orient-decide-act loop

Orient - Since the adversary was acquired first, US pilots could then react by orienting themselves towards the adversary much faster.

Decide - After reacting with their initial orientation, the better level of training then allowed them, as decision makers, to proceed faster to the next combat manoeuvre.

Act - With the next combat manoeuvre decided upon, US pilots could then rapidly input aircraft controls, with the resultant faster initiation of a desired manoeuvre (the F-86 Sabre was more nimble than the MiG-15 because of its fully hydraulic controls).

Boyd conceptualised the principles of the OODA loop in his two famous briefings "patterns of conflict" and "a discourse on winning and losing", which are considered the most dazzling briefings ever to come from a military mind. These presentations began as one-hour and grew to fifteen-hour briefings over two days and were given over 1,500 times. Thousands of copies have penetrated US military and defense circles, particularly at senior levels. Boyd never formally published his observations, but he has been recognized as the greatest military theoretician since Sun Tzu and as the architect of America's strategy in the 1990–1991 Gulf War (Coram 2002, Hammond 2004). The OODA loop has become a standard model of the decision-making cycle not only for the military, but also by many business and research communities around the world (Hammond 2004).

In comparison, Noble Prize winner Herbert Simon studied management decision making and developed a more generalized model of decision making (Simon 1977). Simon's model is shown in Fig. 14.2 with four phases (the final phase added by later researchers) of Intelligence – Design – Choice – Implementation. During the intelligence phase, the user seeks and acquires information needed for the decision problem. Design involves developing criteria important to the decision and establishing relationships between variables of interest. The user makes a selection during choice, and the decision is implemented during the final phase. The phases proceed relatively sequentially, with feedback loops as the user returns to a previous stage before moving forward again. Boyd's model and Simon's model both involve feedback loops and are similar in that the first phase involves acquiring information, the second developing a model to relate the information, the third making a choice, and the fourth acting on the information.

14.3 Intelligent Decision Support Systems

Incorporating AI techniques within DSSs to form IDSSs is not new. However, recent advances have enabled better accessibility to AI technology that has resulted in an increased number of IDSS applications, particularly those using multi-agent systems. These types of applications can aid the decision maker in selecting an appropriate action in real-time under stressful conditions by enabling up-to-date information, reduced information overload,

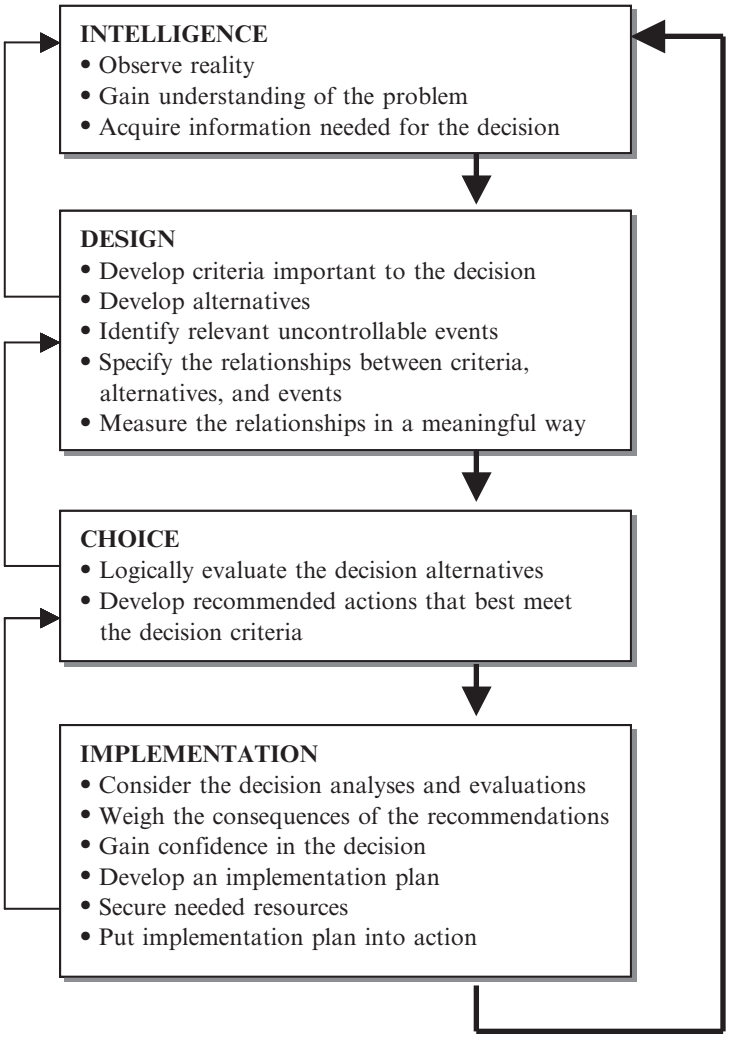

Fig. 14.2. Simon's three phases of decision making, with the last phase added later

and a dynamic response. Intelligent agents can be used to enable communication required for collaborative decisions and to treat uncertainty in the decision problem. AI researchers possess a comprehensive toolbox to deal with issues such as architecture and integration (Mackworth 2005). Several recent examples include:

Petroleum production: Based on Case Based Reasoning (CBR) using bioinformatics (Bichindaritz and Marling 2006, Chan 2005);
Clinical healthcare: Using collaborative decision making and knowledge exchange (Frize et al. 2005);

Forest fire prevention: Based on fuzzy modeling (Iliadis 2005);
Diagnosing breast cancer: Using Linear Genetic programming (LGP);
Multi Expression Programming (MEP) and Gene Expression programming (Jain 2000).

Intelligent Agents (IA) are perhaps the mostly widely applied AI method in IDSSs in recent years due in part to their characteristics of mobility and autonomy. This utilization has significantly advanced many applications, particularly Web-based systems (see for example, Phillips-Wren and Jain 2005). In addition, learning can be incorporated into agent abilities to extend the capability of systems (Valluri and Croson 2005).

14.3.1 Agent Teaming

An agent-enabled IDSS can be designed using a multi-agent system to provide simultaneous data analysis and to enhance the fidelity of feedback to the user. The architecture of the proposed system resembles that of a simple thermostat, containing a monitor and a feedback circuit. Building blocks of this type lead to expert systems and the creation of production rules in the form of logical expressions to embody knowledge. These rules are entered into the knowledge repository as a set of inferences. MYCIN (Simon 1977) and DENDRAL (Feigenbaum et al. 1971) were early commercial versions of DSSs using an expert system as its source of knowledge/inference. An IDSS that uses a multi-agent system to monitor and log the environment prior to deciding on the type and amount of feedback requires significant planning. To interact, the system needs to react to changes at its input from a sensor (using an event-driven model) and produce outputs to drive actuators (again using an event-driven model). These agents can be instantiated using off-the-shelf expert system shells (Negnevitsky 2005). This means that knowledge needs to be represented in terms of rules generated by a subject mater expert prior to use. Such rules should be expressed in terms of Relationships, Recommendations, Directives and Strategies. Separate agents are generally used to collect and refine the test data required to build and test the system. An additional interface agent (or team of agents) is used to interface the inference engine and another agent (or team of agents) to generate feedback and reports.

There are three primary challenges that must be overcome to effectively form agent teams: Communication, Negotiation, and Trust. Communication is concerned with the means of communication between agents such that they can understand each other. Early agent development relied on the idea that intelligence is an emergent property of complex interactions between many simple agents. For example, the Open Agent Architecture (Cheyer and Martin 2001) is based on agents in a community of agents cooperating to achieve their design objectives. Communication between agents must be efficient and robust enough to recover easily from errors, and specialized Facilitator agents are responsible for matching requests with the capabilities of different agents.

Another approach is given by Aglets (Lange 1997) as Java objects that can move from one host on the network to another. Such mobile agents are particularly useful in distributed systems. Finally, Swarm (Group 2005) provides a hierarchical structure that defines a top level observer swarm; a number of model swarm are then created and managed in the level below it.

The second challenge to forming agent teams is Negotiation. Generally, development of teams involves separating the requirements of a team from the requirements of individual agents. This includes assigning goals to the team as a whole, and then allowing the team to figure out how to achieve it autonomously. A team is constructed by defining the number of roles that are required in order to achieve the goals of the team. Additionally, agents can be specifically developed to perform one or more roles. An important feature of this approach is that agents are assigned with roles at runtime and can also change roles dynamically as required. Hence, one agent may need to perform one or more roles during its operation. MadKit (Ferber et al. 2006) is a multi-agent platform built upon an organizational model called Agent/Group/Role, and agents may be developed in many third party languages. The widely used 'JACK Teams' (AOS 2006) provides a team-oriented modeling framework. Specifically, this allows the designer to focus on features such as team functionality, roles, activities, shared knowledge and possible scenarios.

The third major challenge to agent team formation is Trust, specifically how an agent should handle trust in regards to other agents. For example, should an agent trust the information provided by another agent, or trust another agent to perform a particular task. The level of trust is not easily measured, although loyalty can be used to weight information and consequently the strength of bond that is created. The fragility of that bond reflects on the frequency and level of monitoring required for the team to complete the related portion of a task. For further details on trust, the reader may refer to Tweedale and Cutler (2006).

One would expect to gain major benefits from intelligent agent technology through its deployment in complex, distributed applications such as virtual enterprise management and the management of sensor networks. However, while the agent paradigm offers the promise of providing a better framework for conceptualising and implementing these types of systems, there is a need to recognise the underlying programming paradigms and supporting standards, design methodologies and reference architectures needed before these applications can be developed effectively. As noted above, standards are beginning to appear, but more experience and is needed with real applications, and the software community needs to be educated in their use. Given the nature of these applications, a sudden shift to an agreed-upon standard in the community seems unlikely. Rather, as the field matures we would expect to see a gradual shift from object-oriented to the agent paradigm in intelligent domains.

The underlying theories of cognition will continue to prove adequate for large-scale software developments. The key theories (BDI and production systems) date from the 1980s and have a long pedigree in terms of their use

in commercial-strength applications. This longevity indicates that their basic foundation is both sound and extensible as clearly illustrated in the progression of BDI implementations from PRS (Francois et al. 1996) to dMARS (d'Inverno et al. 1997) to JACK (AOS 2005) and to JACK Teams (AOS 2006). New cognitive concepts may gain favour (e.g. norms, obligations, or perhaps commitment), but we believe that these concepts will not require the development of fundamentally new theories.

While we believe that the existing theories are sufficiently flexible to accommodate new cognitive concepts, we perceive a need to develop alternative reasoning models. In the case of the JACK implementation of BDI, a team reasoning model is already commercially available in addition to the original agent reasoning model. On the other end of the spectrum, a low-level cognitive reasoning model (COJACK) has been recently developed. This model enables the memory accesses that are made by a JACK agent to be influenced in a cognitively realistic manner by external behaviour moderators such as caffeine or fatigue. Interestingly, COJACK utilises an ACT-R like theory of cognition, which in turn is implemented using JACK's agent reasoning model. From a software engineering viewpoint, it should be the reasoning model that one employs that shapes an application, not the underlying cognitive theory. There is the opportunity through the provision of "higher level" reasoning models like OODA and their incorporation into design methodologies to significantly impact productivity and, hence, market penetration of these technologies.

14.3.2 Collaborating Agents to Simulate Teamwork

A number of researchers have integrated cognitive decision-making models with agents (Klein 1989b, Yen et al. 2001) to capture the decision making abilities (Klein 1989a) of domain experts based on the recognition of similarity between the current situation and past experiences. In the first (recognition) phase, a decision maker develops situation awareness and decides upon a course of action. In the second (evaluation) phase, a decision maker evaluates each course of action. Klein (1989a, b) introduced a model that evolved into an agent environment under teamwork setting into the Recognition-Primed Decision (RPD) Agent architecture (Fan et al. 2005b). Klein's cognitive model was extensively tested in highly stressful, time-pressured, decision making environments such as those faced by firefighters or military personnel under attack. He proposed that these types of decision makers base their responses on past experience and situations that are similar to the new situation. Hanratty et al. showed an agent architecture for a RPD Agent (Hanratty et al. 2003) as consisting of four modules. The communication manager module governs the inter-agent communication and organises conversations. The expert system module is a rule-based forward chaining system containing knowledge related to the other agents and external world. The process manager module

is responsible for scheduling and execution of plans. The collaborative module facilitates the collaboration between humans and RPD agents (Klein 1989a).

Software called Recognition-Primed Collaborative Agent for Simulating Teamwork (R-CAST) was developed based on the RPD model using similarities between past experience and current situation. The Pennsylvania State University has filed a patent on the software embodied in R-CAST, an extension of the Collaborative Agent for Simulating Teamwork (CAST) architecture (Fan et al. 2005a). CAST was designed to simulate teamwork by supporting proactive information exchange in a dynamic environment, while R-CAST extended CAST architecture with a recognition-primed decision making model. R-CAST consists of a number of modules for handling the collaboration among RPD-agents, between RPD-agent and human, and among humans. The Shared Mental Model (SMM) consists of team processes, team structure, shared domain knowledge, and information-needs graphs. The Individual Mental Model (IMM) stores mental attitudes held by agents. Information is constantly updated using sensor inputs and messages from agents. The Attention Management (AM) module is responsible for the decision-maker agent's attentions on decision tasks. The Process Management (PM) module ensures that all team members follow their intended plans. The functions of the other modules are described by Yen et al. (2001).

The developers of R-CAST and RPD Agent have tested their software in a military command-and control simulation involving intelligence gathering, logistics and force protection (Hanratty et al. 2003). Under normal time pressure, the human teams made correct decisions about the potential threat. As time pressure increases, team performance suffers due to the lack of information sharing resulting in incorrect decisions about whether to attack/avoid the incoming aircraft. The researchers demonstrated that the R-CAST agent systems helped human-agents in making the right decisions under time-pressured conditions. This concept is demonstrated using a scenario in which team members have to protect an airbase and supply route that are under attack by enemy aircraft. The scenarios were configured with different patterns of attack and at different tempos. Two human team members were dependent on a third human whose role was to gather information and communicate to them. The defence teams cannot attack if they do not know whether the incoming aircraft is friend or foe. The supply team takes action to avoid a possible incoming threat. When the information gatherer was supported by the R-CAST software system, the information was processed and shared quickly. As a result, the human-agent teams were able to defend themselves from enemy attack.

14.3.3 JACK Intelligent Agents

JACK Intelligent Agents is a development platform for creating practical reasoning agents in the Java language using BDI reasoning constructs. It allows the designer to use all features of Java as well as a number of specific agent extensions. Any source code written using JACK extensions is automatically

compiled into regular Java code before being executed. Each agent has beliefs about the world, events to respond reactively, goals that it desires to achieve, and plans that define what to do. When an agent is executed, it waits until it is provided with a goal to achieve or receives an event to which it can respond reactively; it then reasons using its beliefs and decides whether to respond. If a response is required, it selects an appropriate plan to execute in order to respond. JACK agents can exhibit: Goal-directed behavior, where the agent focuses on the objective and not the method chosen to achieve it; Context sensitivity, keeping track of which options are applicable at each given moment using beliefs; Validation of approach, ensuring that a chosen course of action is pursued only for as long as applicable; and Concurrency, behaviours in the agent are executed in separate, parallel and prioritized threads.

JACK provides a language for developing agent-based systems using agent-oriented paradigms, and the language is complete with a compiler, a powerful multi-threaded runtime environment and a graphical environment to assist with development. Beliefs have been implemented as relational databases called beliefsets; however, developers can also use their own Java-based data structures if needed. Desires are realized through goal events that are posted in order to initiate reasoning. This is an important feature because it causes the agent to exhibit goal-directed behaviour rather than action-directed behaviour, meaning that the agent commits to the desired outcome and not on the method to achieve it. An intention is defined as a plan to which the agent commits to after choosing from a library of pre-written plans. The agent is able to abort a plan at any time depending on its beliefs and also consider alternative plans.

JACK Teams is an extension to the JACK platform that provides a team-oriented modelling framework. The JACK Teams extension introduces the concept of Team reasoning, where agents encapsulate teaming 'behaviour and roles' required to define what each agent is required to do within the team. Using this Teams extension of JACK, individual agent functionality is also available within a team. Team-oriented programming enables the designer to specify: What functionality a team can perform; What roles are needed in order to form a team; Whether an agent can perform a particular role within a team; Coordination of activities between team members; Knowledge between team members.

Roles are bound to agents at runtime. This means that it is possible to have different combinations of agent-role relationships. For example, on one hand, one role can be performed by many different agents (in which case one agent must be selected at runtime), on the other hand, one agent can also perform many roles simultaneously as required.

Belief propagation allows beliefs to be shared between members of a team. This means it becomes possible for sub-teams to inherit beliefs with important information from higher-level teams and conversely, enclosing teams to synthesize beliefs from lower-level sub-teams. JACK Teams was developed to support structured teams; therefore, the role obligation structure of a team

must be defined at compile-time. Consequently, sub-teams can only communicate and share information if it has been previously defined in their team structure.

14.3.4 Teaming

The team in this concept can initially be considered to consist solely of software agents. However, ultimately the team will include human agents or operators in either a collaborative or commanding mode. The communication aspects in agent teaming address traditional teaming properties such as exchange of information as well as agent and mutual performance monitoring. Research focus is needed in communication and collaboration between software and human agents.

The structure of teams is traditionally defined during the system design and is required to remain constant during operation. Within teams, agents are required to cooperate and explicitly trust other team members. The idea of introducing dynamic, temporary team-like links that can be established or destroyed at runtime also needs to be considered. This approach allows the achievement of greater autonomy since different systems, each executing different agent teams, are able to collaborate in order to achieve their goals. Additionally, agent teaming should be considered to contain a 'human-centric' nature. Current research trends in agent development needs to focus on how agents interact within teams.

One of the major issues in early human-machine automation was a lack of focus on human users and their cognitive processes. Recent developments in intelligent agents have become a popular way to respond to these early deficiencies. Early agent models or theories were attractive solutions due to their human-like intelligence and decision-making behaviour. Existing agent models can act as stand-alone substitutes for humans and their human decision-making behaviours.

At this point we come back to one of the problems in early human-machine automation – the human-like substitute could fail at a critical point due to cultural diversity or lack of coordination, leaving the human no chance to regain control of the situation (usually as a result of impaired situation awareness). A solution was developed by AI researchers who created a machine-assistant operating in an advisory or decision support role and that assisted human operators during critical or high workload situations. This software led to the development of intelligent agent technology. This technology has matured and is now robust enough to implement machine-assistant behaviour (agents that are more independent, co-operative or capable of assisting associates).

Urlings (2003) claims that in order to compose effective human-agent teams and in order to include intelligent agents as effective members in this team, a paradigm shift in intelligent agent development is required similar to the change from the technology-driven approach to the human-centered approach in automation. He provides an example based on the operational

analysis domain. He proposes that the traditional development of agent technology failed to distinguish between a software agent and a human, preventing them from being interchangeable, even though they are 'inherently different'. By establishing the difference between agents and humans, Urlings states that in a typical human-agent team both entities are not comparable but are complementary to each other by means of cooperative sharing of tasks while working in concert.

This work on first principles of human-centered automation is explained as follows: Humans are responsible for outcomes in human-agent teams; The human must therefore be in command of the human-agent team; To be in command, the human must be actively involved in the team process; To remain involved, the human must be adequately informed; The human must be informed about (able to monitor) agent behavior; The activities of the agents must therefore be predictable; The agents must also be able to monitor performance of the human; Each team member (humans and agents) must have knowledge of the intent of the other members (Urlings 2003).

We believe that human-centric agents could benefit from human cognition theories as an extension of their inherent reasoning. Researchers have demonstrated that teams can work effectively using a shared mental model, and R-CAST offers a promising technique for human-agent collaboration. A number of researchers in the multi-agent community are developing human-machine teaming systems for use in difficult and critical decision making under high workload situations. Human-machines teams are still led by humans, but we expect that human-control will be slowly transferred to machine-control as machines become autonomous and intelligent.

14.4 The Human-Centric Approach

In order to understand where human-agent collaboration fits into current agent trends, we need to have a close look at the classification of agents. We think that one such classification provides an accurate description of current agent trends. Nwana (1996) chooses to classify agent topology using categories such as mobility, reasoning, autonomy and hybrid.

Agents may have characteristics from multiple categories. For example mobile agents can posses learning attributes. Here we will focus on the third category since it is the leading area of current research in agents as well as the foundation needed for Teaming (coordination and cooperation). In this category, autonomy represents 'taking initiative' instead of simple responsive action towards the environment. Cooperation represents the 'interaction' needed to form intelligence, and the key element of intelligence is 'learning'. Nwana (1996) extends these three ideal attributes with the integration of the other categories. The resulting overlap in characteristics produces purely collaborative agents, collaborative learning agents, interface agents and ultimately smart agents. Purely collaborative agents are autonomous entities

that coordinate their activities while not necessarily collaborating with other agents (proactively collaborating activities). Collaborative learning agents are self-performance improving (learning by observation) agents by observing others (agents or humans). Interface agents' typologies emphasize autonomy and learning, giving rise to application areas such as support and assistance to a user by adapting to the specific skill set so that the user 'feels' comfortable. Finally a 'smart agent' as described by Nwana (1996) should learn and interact with its external environment.

Reasoning models of agents play an important part in their existence; they have been categorized as deliberative and reactive. Purely reactive reasoning is very much like stimulus-response type, where the action is chosen based on previously defined action-response pairs. Reactive agents are most suited to less dynamic environments and for quicker response in real-time. On the other hand, deliberative reasoning is inspired from cognition theories and imitates human-like reasoning in agents. Deliberative reasoning is generally slower than reactive reasoning, but it has advantages of giving more human-like intelligence. This was one of the reasons why the early deliberative agent paradigms such as BDI became popular and widely-accepted in the agent community.

Although the BDI paradigm is widely used to mimic human intelligence, BDI agents can not be fitted in to the above definition of truly 'smart agents' since they still lack the primary ideal characteristics of 'Coordination and Learning'. We expect that one of the major steps of the next generation of agents will comprise coordination (Teaming) and, ultimately, learning.

We think that another major step in agent teaming research will be to introduce a 'human-centric' nature within an agent's architecture. The current trend in agent development is focused on its agent-only interaction, meaning that agent teaming is comprised of joint-goal operations that consist of agents as sole subordinates of the team without any human intervention. Here we distinguish between the need of a human in the loop as a colleague and as a sometimes supervisory role. This demands agent architectures to embody social ability in order to interact with the human part of the team. In Hopkins and DuBois (2005), Wooldridge describes social ability as "the ability to interact with other agents and possibly humans via some communication language."

We would like to suggest that 'interaction' with humans cannot only be via some communication language, but also can be by other means such as observation and adaptation. We would also like to suggest that truly smart agents can be complementary to a human by adopting skills similar to a human (and that may include communication, learning and coordination) rather than being a simple replacement to a human. Such a view encourages research focused on developing the agent's human-centric nature by combining one or more ideal attributes such as coordination, learning and autonomy.

14.5 Steps Toward Next Generation

The BDI agent model has the potential to be a method of choice for complex reactive systems. Future trends in agent technology can be categorized on the basis of 'Teaming' which can be divided into Multi-Agents (Teaming) and Human-Centric Agent (Human-Teaming). These two research streams have two commonalities, namely, collaboration and cooperation. Along with these, a human-centric agent possesses ideal attributes such as learning as discussed previously in the definition of a truly smart agent. Recent work on the BDI agent such as Shared Plans/Joint Intentions and JACK teams (AOS 2004) facilitates agent-only teaming. Furthermore, the addition of an ideal attribute such as learning enables agents to come closer to the goal of a human-centric smart agent.

Agent collaboration provides the opportunity for agents to share resources during their execution. Such resources are not normally available within current multi-agent system designs because resources are allocated for the use of specific teams. Team structures and team members are defined explicitly when the system is being designed. Using collaboration, agents are able to recognize when additional resources are needed and negotiate with other teams to obtain them. Collaboration is a natural way to implement human-agent teaming due the temporary and unpredictable nature of human team members.

14.6 Building a Teaming Framework

The case study presented in this section describes the proposed first steps in understanding how to implement human-agent teaming in an intelligent environment. A prototype implementation framework has been developed that allows an agent to establish collaboration with another agent or human. The framework is based on CHRIS (Sioutis and Ichalkaranje 2005), an agent reasoning and learning framework developed as an extension of JACK at the University of South Australia (Sioutis 2006). CHRIS equips a JACK agent with the ability to learn from actions that it takes within its environment. It segments the agent reasoning process into five stages based on a combination of functions extracted from Boyd's OODA loop (Hammond 2004), Rasmussen's Decision Ladder (Sioutis et al. 2003) and the BDI model (Rao and George 1995). Boyd's Orientation stage has been implemented as a collaboration module, which itself has been limited between the State and the Identification operation as shown in Fig. 14.3.

The path on the left shows the process involved in establishing a collaboration contract between two or more agents. The path on the right indicates that agents need to continuously perform assessment in order to ensure that collaboration is progressing as previously agreed upon and also whether the collaboration is yielding the required effect toward achieving each agent's

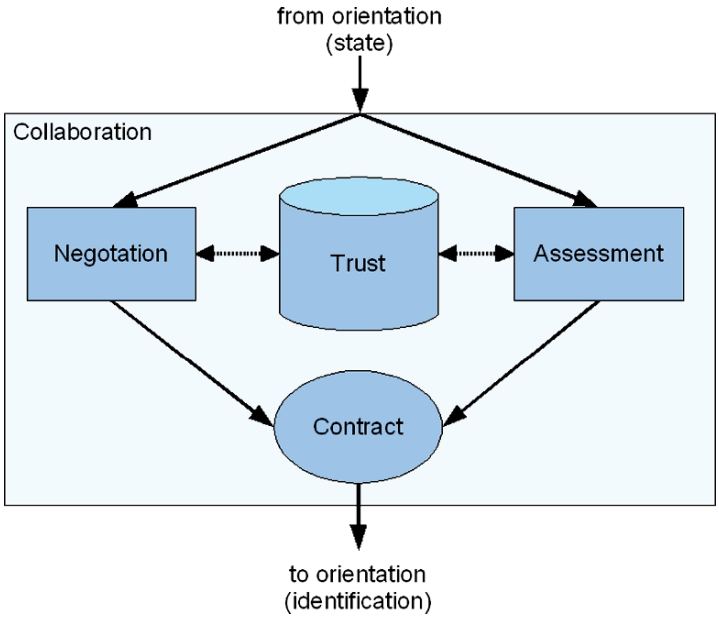

Fig. 14.3. Reasoning and collaboration

own goals. Both of these operations are highly dependent on trust, which is updated accordingly.

This implementation is based on using JACK team agents. Negotiation is performed using an authoritative Collaboration Manager Agent. Subordinate agents simply need to be able to perform the Cooperation role. The current implementation only supports goal-based collaboration relationships, where an agent negotiates for another agent to achieve a particular goal. Finally, an event called RequestCollaboration is used to ask the Collaboration Manager Agent for collaboration.

14.6.1 Decision Making Using a Human-Agent Team

A demonstration program was written that provides limited human-agent collaboration. It uses two agents. The first agent called Troop connects to a computer game called Unreal Tournament (UT) using UtJackInterface (Sioutis 2003) and controls a player within the game. The second agent is called HumanManager and is used to facilitate communication with humans encountered within the game. The program demonstrates how the Troop agent is given the goal hierarchy shown in Fig. 14.4. This Decision Making Agent is used to decide whether the entity will Defend or Attack. The Troop agent can only perform the Defend or Attack goal (mutually exclusive). This agent decides how to handle the Attack goal and then asks the Collaboration Manager Agent to organise other (friendly) human players encountered

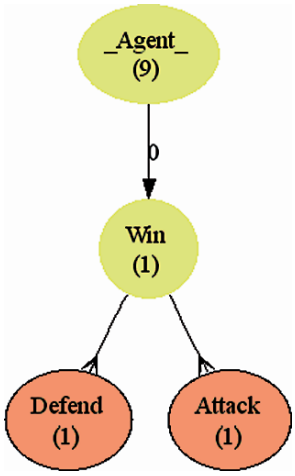

Fig. 14.4. Goal hierarchy used for demonstration

in the game to take responsibility for the alternate goal. The sequence of operations for the demonstrations is:

1. The agents Troop, Collaboration Manager Agent and HumanManager are created and a scenario.def file is used to form a team with the Cooperation role between the Collaboration Manager Agent and the HumanManager.
2. The Win goal is activated and the Defend and Attack sub-goals are subsequently activated automatically in parallel. Attack is handled by the Troop agent that subsequently attacks any enemy that comes within the field of view. For demonstration purposes, the Attack goal succeeds after the agent attacks five enemy players.
3. A RequestCollaboration message is sent to the Collaboration Manager Agent for the Defend goal. The Collaboration Manager Agent then executes an @team achieve for any sub-teams that perform the Cooperation role. The HumanManager agent then negotiates and performs assessment with the human in order to satisfy the Defend goal.

The human's point of view is acknowledged by:

(a) Asking the Human
(b) The Human Refuses
(c) The Human Accepts

The human is able to communicate with agents via text messages through UT. Figure 5a illustrates what appears on the human's monitor when a message is received from the agent. The sequence diagram shown in Fig. 5b illustrates that if the human refuses to join the team, the collaboration fails and hence both the Defend and Win goals both fail. On the other hand, the sequence diagram shown in Fig. 5b illustrates that when a human accepts to

Fig. 5a. The CMA model of a human's decision cycle (Asking the human)

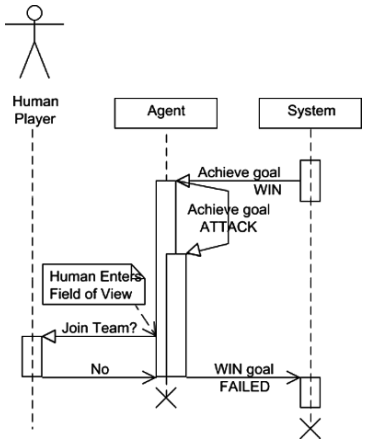

Fig. 5b. The CMA model of a human's decision cycle (The human refuses)

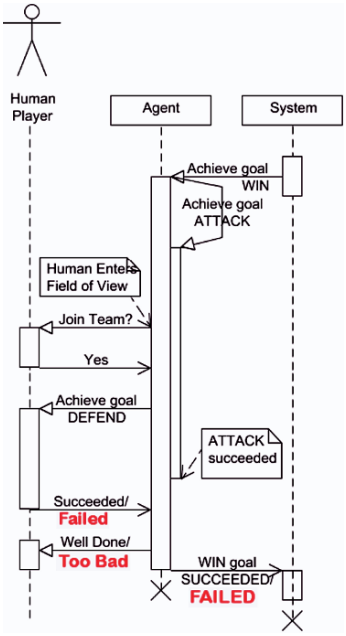

Fig. 5c. The CMA model of a human's decision cycle (The human accepts)

join the team, collaboration is formed and the human is assigned with the Defend goal. The result of the Win goal then depends on whether the human reports that he/she was successful in achieving the Defend goal.

14.7 Concluding Remarks

Intelligent agent technology is at an interesting point in its development (Valluri and Croson 2005). Commercial-strength agent applications are increasingly being developed in domains as diverse as meteorology, manufacturing, war gaming, capability assessment and UAV mission management. Furthermore, commercially-supported development environments are available and design methodologies, reference architectures and standards are beginning to appear. These are all strong indicators of a mature technology. However, the adoption of the technology is not as rapid or as pervasive as its advocates have expected. Intelligent agent technology has been promoted as the paradigm of choice for the development of complex distributed systems and as the natural progression from object-oriented programming. Is intelligent agent technology simply in need of a 'killer application' for demonstration, or are there more fundamental reasons as to why a technology that promises so much has not been more widely adopted? What does the future hold for this technology?

The development of intelligent agent applications using current generation agents is not yet routine. Certainly providing more intuitive reasoning models and better support frameworks will help, but we see behaviour acquisition as a major impediment to the widespread application of the intelligent agent paradigm. The distinguishing feature of the paradigm is that an agent can have autonomy over its execution, i.e. an intelligent agent has the ability to determine how it should respond to requests for its services. This is contrasted with the object paradigm, where there is no notion of autonomy and objects directly invoke the services that they require from other objects. Depending on the application, acquiring the behaviours necessary to achieve the required degree of autonomous operation can be a major undertaking. The problem can be likened to the knowledge acquisition bottleneck that beset the expert systems of the 1980s. Thus, there is a need for principled approaches to behaviour acquisition, particularly when agents are to be deployed in behaviour-rich applications such as enterprise management. Cognitive Work Analysis has shown promise in this regard, but further studies are required.

Alternatively, the requirement for autonomous operation can be weakened and a requirement for human interaction introduced. Rather than having purely agent-based applications, cooperative applications involving teams of agents and humans could be developed. Agent-based advisory systems can be seen as a special case of cooperative applications, but we see the interaction operating in both directions, i.e. the agent advises the human, but the human also directs and influences the reasoning processes of the agent. Existing architectures provide little in the way of support for this two-way interaction. Such

interactions require that the goals and intentions of both the human and the agent are explicitly represented and accessible, as well as the beliefs that they have relating to the situation. This approach provides a convenient way to address the difficulties associated with behaviour acquisition associated with autonomous operation. By making the agent's longer term goals and intentions visible, as well as the rationale behind its immediate recommendation, this approach also provides a mechanism for building trust between humans and agents. It should also be noted that in many applications, such as cockpit automation and military decision making, full autonomy is not desirable; an agent can provide advice, but a human must actually make the decision. In these cases, we expect to see an increasing number of applications designed specifically for human teams, agent teams or a combination of both.

Learning has an important role to play in both cooperative and autonomous systems. However, the reality is that learning is extremely difficult to achieve in a general and efficient way, particularly when dealing with behaviours. The alternative is to provide the agent with predefined behaviours based on à priori knowledge of the system and modified manually from experience gained with the system. This has worked well in practice and we expect that it will remain the status quo for the immediate future.

In summary, we expect that intelligent agents will retain their architectural foundations but that the availability of more appropriate reasoning models and better design methodologies will see them increasingly used in mainstream software development. Furthermore, better support for human-agent teams will provide the impetus for the development of a new class of intelligent decision support applications (Tweedale et al. 2007).

Acknowledgements

Some of our research results presented in this chapter also appear in the proceedings of the 10th KES Conference, 2006, published by Springer and the International Journal of Decision Support Technologies, IOS Press, Volume 1, Number 4, 2007.

References

AOS (2004) JACK Intelligent Agents: Teams Manual 3.0, Agent Oriented Systems, Melbourne.

AOS (2005) What is jack? [Online, accessed 6 June 2005] http://www.agent-software.com.au/shared/products/index.html.

AOS (2006) Jack intelligent agents teams manual 4.1, [Online, accessed 2d March] http://www.agent-software.com.au/shared/resources/index.html.

Bichindaritz, I. and Marling, C. (2006) Case-based reasoning in the health sciences: What's next? Artificial Intelligence in Medicine 36, pp. 127–135.

Bratman, M. E. (1999) Intention, plans, and practical reason, center for the study of language and information.

Chan, C. (2005) An expert decision support system for monitoring and diagnosis of petroleum production and separation processes, Expert Systems with Applications 29, pp. 127–135.

Cheyer, A. and Martin, D. (2001) The open agent architecture, Journal of Autonomous Agents and Multi-Agent Systems 4(1), pp. 143–148.

Coram, R. (2002) Boyd: The Fighter Pilot Who Changed the Art of War, Little Brown and Company, Boston.

Curts, R.J. and Campbell, D.E. (2001) Avoiding information overload through the understanding of OODA loops, a cognitive hierarchy and object-oriented analysis and design, in Proceedings of the 6th International Command and Control Research and Technology Symposium (CCRTS), US Naval Academy, Annapolis, MD, pp. 312–319.

d'Inverno, M., Kinny, D., Luck, M. and Wooldridge, M. (1997) A formal specification of dMars, in Agent Theories, Architectures, and Languages, pp. 155–176.

Fan, X., Sun, S. and Yen, J. (2005a) On shared situation awareness for supporting human decision-making teams, in 2005 AAAI Spring Symposium on AI Technologies for Homeland Security, Stanford, CA, pp. 17–24.

Fan, X., Sun, S., Sun, B., Airy, G., McNeese, M. and Yen, J. (2005b) Collaborative RPD-enabled agents assisting the three-block challenge in command and control in complex and urban terrain, in 2005 Conference on Behaviour Representation in Modelling and Simulation (BRIMS), Universal City, CA, pp. 113–123.

Feigenbaum, E.A., Buchanan, B. and Lederberg, J. (1971) On generality and problem solving: a case study using DENDRAL program, in Edinburgh, B., Scotland, M. and Michie, D. eds, Machine Intelligence, Vol. 6, Edinburgh University Press, pp. 165–190.

Ferber, J., Gutkecht, O. and Michel, F. (2006) Madkit development guide, [Online accessed: 25 August 2006] http://www.madkit.org/madkit/doc/devguide/devguide.html.

Francois, F.I., Chatila, R., Alami, R. and Robert, F. (1996) PRS: A high level supervision and control language for autonomous mobile robots, in Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, USA, Vol. 1, IEEE Press, pp. 43–49.

Frize, M., Yang, L., Walker, R. and O'Connor, A. (2005) Conceptual framework of knowledge management for ethical decision-making support in neonatal intensive care, IEEE Transactions on Information Technology in Biomedicine 9, pp. 205–215.

Group, S.D. (2005) Documentation set for swarm 2.2, [Online Accessed: 25 August 2005] http://www.swarm.org/swarmdocs-2.2/set/set.html.

Hammond, G.T. (2004) The Mind of War: John Boyd and American Security, Smithsonian Institution Press: Washington, USA.

Hanratty, T., Dumer, J., Yen, J. and Fan, X. (July 2003) Using agents with shared mental model to support objective force, in Cybernetics and Informatics (SCI 2003), 7th World Multiconference on Systemics, Orlando, USA, pp. 27–30.

Hopkins, M. and DuBois, C. (2005) New software can help people make better decisions in time-stressed situations, Science Daily pp. 1–2.

Iliadis, L. (2005) A decision support system applying an integrated fuzzy model for long-term forest fire risk estimation, Environmental Modelling and Software 20, pp. 613–621.

Jain, A., et al. (2000) Artificial Intelligence Techniques in Breast Cancer Diagnosis and Prognosis, World Scientific Publishing Company, Singapore.

Klein, G.A. (1989a) Recognition-primed decision (RPD) model of rapid decision making, IEEE Transactions on Systems, Man and Cybernetics 19, pp. 462–472.

Klein, G.A. (1989b) Recognition-primed decisions, Advances in Man-Machine System Research 5, pp. 47–92.

Klein, G.A. and Crandell, B.W. (1995) The role of simulation in problem solving and decision making.

Lange, D.B. (1997) Java Aglet Application Programming Interface (J-AAPI) White Paper - Draft 2, Technical report, IBM Tokyo Research Laboratory.

Mackworth, A. (2005) The coevolution of AI and AAAI, AI Magazine 26, 51–52.

Negnevitsky, M. (2005) Artificial Intelligence, A Guide to Intelligent Systems, Pearson Education Limited, Harlow, England.

Nwana, H.S. (1996) Software Agents: An overview, Knowledge Engineering Review 11(3), pp. 1–40.

Phillips-Wren, G. and Jain, L. eds (2005) Intelligent Decision Support Systems in Agent-Mediated Environments, IOS Press, The Netherlands.

Rao, A. and George, M. (1995) BDI Agents: from Theory to Practice, in Proceedings for the 1st International Conference on Multi-Agent Systems (ICMAS-95), AAAI Press, California, pp. 312–319.

Russel, S. and Norvig, P. (2003) Artificial Intelligence: A Modern Approach, second edn, Prentice-Hall, Inc., Englewood Cliffs, N.J. 36. Shortlife, E. (1976) MYCIN: Computer-Based Medical Consultations, Elsevier New York, NY.

Simon, H.A. (1977) The New science of Management Decision (3rd edition revised, first edition 1960), Prentice-Hall, Englewood Cliffs, NJ.

Sioutis, C. (2006) Reasoning and Learning for Intelligent Agents, PhD thesis, School of Electrical and Information Engineering, University of South Australia.

Sioutis, C. and Ichalkaranje, N. (2005) Cognitive hybrid reasoning intelligent agent system, in R. Khosla, R.J. Howlett and L.C. Jain, eds, Proceedings of the 9th International Conference on Knowledge Based Intelligent Information and Engineering Systems (KES 2005), Sep. 14–16, Melbourne, Australia, Vol. 3681(2) of Lecture Notes in Artificial Intelligence, Springer, Berlin Heidelberg New York, pp. 838–843.

Sioutis, C., Ichalkaranje, N. and Jain, L. (2003) A framework for interfacing BDI agents to a real-time simulated environment, in A. Abraham, M. Koppen and K. Franke eds, Design and Application of Hybrid Intelligent Systems, Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, The Netherlands, pp. 743–748.

Tweedale, J. and Cutler, P. (2006) Trust in multi-agent systems, in Proceedings of the 10th International Conference on Knowledge Based Intelligent Information and Engineering Systems (KES 2006) edited by Howlett, R.P., Gabrys, B. and Jain, L.C., Oct. 9–11, Bournemouth, England, Vol. 4252 of Lecture Notes in Computer Science, Springer Berlin Heidelberg New York, pp. 479–485.

Tweedale, J., Ichalkaranje, N., Sioutis, C., Urlings, P. and Jain, L.C. (2007b) Building a decision-making framework using agent teams, Intelligent decision technologies: An International Journal, IOS Press 1(4) (in Press.).

Urlings, P. (2003) Teaming Human and Machine, PhD thesis, School of Electrical and Information Engineering, University of South Australia.
Valluri, A. and Croson, D. (2005) Agent learning in supplier selection models, Decision Support Systems 39, pp. 219–240.
Wooldridge, M. (2002) An Introduction to Multi-Agent Systems, Wiley, Chichester.
Yen, J., Yin, J., Ioerger, T.R., Miller, M.S., Xu, D. and Volz, R.A. (2001) Cast: collaborative agents for simulating teamwork, IJCAI, pp. 1135–1144.