BDI-based Human Decision-Making Model in Automated Manufacturing Systems
Xiaobing Zhao, Young-Jun Son*
Systems and Industrial Engineering, The University of Arizona, Tucson, AZ, USA
*Telephone: 1-520-626-9530, E-mail: son@sie.arizona.edu

Abstract
Advances in automation technologies changed the manufacturing environment, where equipment replaced many physical functions of a human. However, humans are still imperative in the automated manufacturing system due to their decision-making functions. This paper presents a novel software agent model to replace the partial decision-making function of a human and its implementation in a distributed computing environment. The proposed model, employing the BDI (belief, desire, intention) agent framework, is capable of 1) generating a plan in real-time, 2) supporting both the reactive as well as proactive decision-making, 3) maintaining situation awareness in human language like logic to facilitate real human decision-making, and 4) changing the commitment strategy adaptive to historical performance. To this end, the intention module in the traditional BDI framework is expanded to include 1) deliberator, 2) planner (reasoning processor), and 3) decision-executor sub-modules. Furthermore, a confidence state is also considered, which affects as well as is affected by three other mental modules (BDI). In this paper, the proposed model has been developed in the context of the human operator who is responsible for error detection and recovery in a complex automated manufacturing system. To this end, the sub-functions required for error detection and recovery are formally mapped on to the beliefs, desires and intentions, and LORA logic is employed to represent them. A scheme of integrating the proposed human agent with an automated shop floor control system (environment) is also developed to demonstrate the proposed agent in an automated manufacturing system context. The model has been implemented in JACK software; since JACK does not support real-time generation of a plan, the planner sub-module is developed in Java. To facilitate integration of an agent (JACK), real human, and the environment (Arena simulation software), a distributed computing platform based on DOD High Level Architecture (now IEEE 1516 Standard) has been used.
Keywords
Human decision-making model, BDI agent, LORA, error detection and recovery, automated manufacturing systems

1. Introduction

Advances in automation technologies have enabled automated equipment to perform many physical functions (e.g. mechanical functions, coordination of equipment, or arithmetic calculations) more efficiently, reliably and economically than humans in automated systems (e.g. automated manufacturing systems) [Grier and Hsiao, 1990; Philips, 1985]. However, the current automated systems lack ingenuity, creativity, and reasoning abilities of humans, and may fail seriously under unanticipated circumstances. Hence, humans are still imperative and often act as a supervisor or a monitor to ensure the functioning of the automated system.

When a human acts as a supervisor or a monitor of an automated system, he or she has to maintain a clear situation awareness to make an informed decision. Unfortunately, increasing automation (due to embedded, invisible functions) makes it difficult for the human to keep track of the activities of the system. Furthermore, humans are subject to inconsistencies and forgetfulness and allow their cognitive functions to disengage without realizing it [Haight and Kecojevic, 2004]. As a result, humans are often not able to obtain the appropriate information about the system’s intentions or actions soon enough to allow them to take corrective actions to avoid a system upset or incident [Haight and Kecojevic, 2004].

As discussed above, while humans’ decision-making abilities are imperative to the functioning of an automated system, they are unable to maintain timely situation awareness, subject to inconsistencies, and prone to errors. To resolve this dilemma, this paper presents a novel software agent to replace the partial decision-making function of a human, whose role involves only decision-making functions as opposed to physical functions. The proposed agent is configured so that it operates autonomously until it faces a situation that cannot be handled by itself, in which case it asks for a human’s input. To this end, there are two detailed objectives.

The first objective is to develop a human decision-making model (based on which the agent will be constructed), which is capable of 1) generating a plan (a sequence of actions required to achieve a goal) in real-time as opposed to selecting a plan based on a static algorithm and predefined/static plan templates that have been generated off-line, 2) supporting both the
reactive as well as proactive decision-making, 3) maintaining situation awareness in human language like logic to facilitate real human decision-making (in the case the agent cannot handle the situation), and 4) changing the commitment strategy adaptive to historical performance. In this work, our model is developed enhancing the traditional BDI (belief, desire, and intention) framework. The real-time planning feature is important because the goal of the system may change dynamically, especially under today’s rapidly changing environment. The reactivity feature allows the proposed agent to respond to critical circumstances (such as alarms) in a timely manner to avoid possible further damages [Ingrand and Georgeff, 1992]; alternatively, the proactiveness feature allows the agent to applying goal-oriented reasoning in performing intellectual tasks. The agent also maintains situation awareness in human language like logic (LORA logic in this paper) to facilitate real human decision-making (in the case the agent cannot handle the situation). Finally, the proposed agent evaluates its performance in real-time to select a commitment strategy to be adopted in the future. Depending on the commitment strategy, the decision-making mechanism is different, which will be discussed later in this paper (see Section 3.4.3).

The second objective is to develop a human model in the context of the human operator who is responsible for error detection and recovery in a complex automated shop floor control system. To this end, the sub-functions required for error detection and recovery are formally mapped on to the beliefs, desires and intentions (BDI). In this work, we have employed LORA logic (Wooldrige, 2000) to represent beliefs, desires, intentions, and plans. We have picked LORA logic because of its rich expressiveness, combining first-order logic, temporal logic, and modal logic.

The third objective is to develop a scheme of integrating the proposed human agent with an automated shop floor control system (environment). To demonstrate the proposed agent in an automated manufacturing system context, this work is essential. The contribution of this work to the field of automated shop floor control is significant, allowing seamless integration among the control system, an agent, and real-human.

The fourth objective is to implement the proposed human model in JACK (state-of-the-art intelligent agent software). Furthermore, since JACK does not support real-time generation of a plan (it only supports static/predefined plans), another objective is to develop the planner sub-module in Java, which is callable from JACK software. The developed agent has been
tested in a simulated environment (in Arena discrete event simulation package) using the distributed computing platform that we have developed (see below).

The fifth objective is to develop a distributed computing platform, in which the agent, real human, and the environment (real or simulated) can be easily integrated to enable agent-in-the-loop, human-in-the-loop system (real system or simulated system). The proposed platform is flexible, allowing us to test and demonstrate alternative agents, humans, and environments.

2. Background and Brief Literature Review

Modeling of human decision-making for a general situation is extremely difficult, if not impossible. However, it becomes more tractable in the context of a specific situation or activity (e.g. manufacturing systems) [Jones and Mitchell, 2002]. The intelligent agent [Norling, 2003], is a promising technique to model a human in a situated, automated system. An intelligent agent is situated within an environment and acts autonomously within that environment, and people are the archetype for autonomous action [Fikes and Nilsson, 1971]. An intelligence agent has its own characteristics such as autonomous, social ability, reactivity, pro-activeness, cooperative, learnable and adaptable [Laughery, 1998]. Autonomy means that the agent operates without direct, continuous supervision. Reactivity means the agent has perceptions of the world inside which it is acting and reacts to changes timely. Pro-activeness means that its behavior is not exclusively reactive but also driven by internal goals.

The best-known and most mature type of intelligence agent is the so-called belief-desire-intention (BDI) model. BDI agent is characterized by its “mental state” with three components: beliefs, desires, and intentions [Rao and Georgeff, 1998]. Beliefs correspond to information the agent has about the world. It may be incomplete or incorrect. Desires represent state of affairs that the agent would wish to be brought about. Intentions represent desires that the agent has committed to achieve. The BDI agent has several merits. First, the BDI paradigm is based on folk psychology, where the core concepts of the paradigm map easily to the language people use to describe their reasoning and actions in every life [Norling, 2004]. This characteristic allows the BDI paradigm-based system to imitate the human decision-making process and to be easily understood by the real human. Second, there exists a formal logic (LORA) to represent the framework of the BDI paradigm. Third, the BDI paradigm is a relatively mature framework and has been successfully used in a number of medium to large scale software systems. Forth,
several software packages, including such as AgentSpeak, Jack, and Jadex, exist to support the modeling process. Finally, the BDI agent can be easily integrated with other agent-based systems and/or other complex, automated systems.

Grier and Hsiao [1990] noticed that the lack of intelligence in the highly automatic manufacturing systems is a critical problem and proposed an expert system to correct it. Discrete event simulation can describe the dynamic behaviors of a system and the interactions among entities in the system in detail. The environment the operator faces can be clearly defined with simulation, and if combined with the expert system simulation also can be applied to depict the human decision-making process. Robinson et al. [2001] proposed to model the human decision-making with simulation. Laughery [1998] also proposed to use discrete event simulation and task modeling technique to model the human performance. The shortcomings of the expert system are that the knowledge acquisition task is time-consuming, and it can not identify new fault type (unforeseen errors) nor cover “neighboring faults” [McDonald et al., 1996].

Mitchell [1987] proposed operator function model (OFM), a heterarchic-hierarchic network of nodes that represent how a human operator manages multiple concurrent activities in a dynamic, event-driven world. Nodes at the top level represent major operator functions, and each function is decomposed into a collection of sub-functions. Each sub-function is in turn composed of a number of tasks, and each task is accomplished by one or more operator actions [Jones et al., 1995]. The OFM is proposed as a modeling tool that provides a dynamic, task-analytic structure that can be used by system designers to define a user interface that is based on operator rather than hardware function [Mitchell, 1987]. The OFM model has been widely used to model the human operator’s major functions, such as control, scheduling, and decision making [Jones and Mitchell, 1994; Ammons et al., 1988]. While OFM is successful on the modeling of the functions of the human operator in a dynamic system, it can not be used to model the human goal-oriented reasoning ability.

In the BDI agent area, Kinny et al. [1996] presented a methodology and modeling technique to describe the external and internal perspective of multi-agent systems based on BDI architecture, and illustrated the approach using an air-traffic management system. Norling [2003] used JACK to encode the knowledge elicitation methodology that mapped closely to the philosophical underpinnings of BDI. Rao and Goergeff [1995] integrated the theoretical foundations of BDI agents from both a quantitative decision-theoretical perspective and a
symbolic reasoning perspective, and also discussed the practical application problems. All these research efforts offered us valuable experiences to develop our BDI paradigm based human model.

In the distributed computing area, the use of multiple models in analyzing complex systems requires an effective interfacing and interoperability of these models. Several works have been presented in describing architectural framework, communication requirements, protocols and algorithms for coordinating and interfacing the multiple models (e.g. simulations) [Kacuter et al., 1998; Fujimoto, 2000; Zeigler and Sarjoughian, 2000; Taylor et al., 2002]. The High Level Architecture (HLA) has become the de-facto standard in distributed simulation [Kuhl et al., 1999; McLean and Riddick, 2000]. Venkateswaran and Son [2004] have addressed the application of distributed simulation technology to evaluate potential supply chains. They enabled distributed simulation of multiple DES models, each representing a member of the supply chain, by interfacing the models using HLA Run Time Infrastructure (RTI). Later, Venkateswaran et al. [2005] extended their work to integrate nonlinear optimization models, system dynamics models, and discrete event simulation models. In this paper, we extend the distributed computing infrastructure proposed in Venkateswaran et al. [2005] to integrate the agent, real human, and the environment (real or simulated).

3. Proposed Human Agent Model

3.1 Human Operator in Automated Manufacturing System

It is noted that beliefs, desires, intentions, and plans are specific to a role of human of particular interest. In this paper, a human model is developed in the context of the human operator who is responsible for error detection and recovery in a complex automated shop floor control system (see Figure 1). It should be noted that the same modeling framework will be applicable to the human operators dealing with complex systems in Air Force (e.g. pilots) and in civilian systems such as operators in a nuclear reactor, power plant, and extended manufacturing enterprise.

In order to construct a formal model of a human, the human’s role is first to be defined. In this work, a human’s role in an automated manufacturing systems is defined by identifying his/her responsibilities and services required to fulfill those responsibilities (See Figure 2). As shown in Figure 2, a human’s role may include the responsibilities of monitoring and error
recovery, making real-time scheduling decisions, processing information, among others. These responsibilities are fulfilled by collecting corresponding information, goal-oriented reasoning, executing commands, among others. As mentioned before, monitoring and error recovery is of our interest in this paper.

![Figure 1. Human operator in an automated manufacturing system](image)

### Figure 1. Human operator in an automated manufacturing system

**ROLE**

**RESPONSIBILITIES**

- Monitoring and Error Recovery
- Processing Information
- Scheduling in Real-time
- Making Decisions
- Handling Materials

**SERVICES**

- Diagnosing and Debugging
- Executing Command
- Communicating
- Collecting Information
- Goal-Oriented Reasoning
- Comparing Performance with Target
- Fine-tuning the Process
- Providing Mechanical Functions

![Figure 2. Role of human in automated manufacturing systems](image)

### Figure 2. Role of human in automated manufacturing systems

#### 3.2 Overview of the Proposed Human Agent Model

The overview of the proposed human agent model is shown in Figure 3, where the model is developed enhancing the traditional BDI (belief, desire, intention) framework. More specifically, the intention module in the traditional framework is expanded to include 1) *deliberator* sub-module, 2) *planner* (also called reasoning processor) sub-module, and 3) *decision executor* sub-module. Furthermore, an *confidence state* is also considered in the model,
which affects as well as is affected by three other mental modules (belief module, desire module, and decision-making module). The proposed human model (see Figure 3) works according to the following steps. First, the agent (model) continually observes the environment to update its initial beliefs via the *perceptual processor* in the belief module. The data about the human himself/herself is also stored. The observed data from sensors and the human himself/herself are in different formats, such as visual data, auditory data, and vibration data. The perceptual processor transforms these heterogeneous data into a standard format called *beliefs*. In this work, a simplified LORA (Logic of Rational Agents) is employed to represent the beliefs. The perceptual processor then filters the data subjectively to obtain information related to the human’s responsibilities and services. As a result, the human has only partial and possibly biased information about the environment and himself. Hence, it is referred to as *beliefs*, and not knowledge. The *beliefs* are updated as the system involves.

![Figure 3. Overview of the proposed human model](image)

Second, based on the current beliefs and initial intentions, the agent decides what states of affairs to achieve (*desires*) via the cognitive processor. The desires are also stored in LORA format. The desires are updated as the system evolves. It is noted that a human can inherit
desires of another system. For example, when the human works as a monitor in an automated manufacturing system, he/she can inherit the desires of the automated manufacturing system. Thus, by comparing its current beliefs with its desires (also the desires of the automated system), the human monitors the progress of the automated system. This will be discussed in more detail in Section 6.

Third, the agent then filters these desires and selects some of them (intention) to commit to via the deliberator. The agent then generates alternative plans via human like goal-oriented planning (planner) based on the current beliefs and guided by intentions. A plan in this paper is defined as a sequence of ordered actions that the agent needs to perform to achieve its intentions. If the planner cannot solve the problem, the agent will post the problem to a real human. Finally, the agent selects an optimal/satisfactory plan based on decision-models and executes it via the decision executor.

The confidence index denotes the human’s attitude. For example, if the human performs some major actions or plans successfully, he/she will be in the confident mode. Otherwise, he/she will be disappointed and will be in the suspicious mode. An equation for the confidence index and a threshold value to switch between the confident and suspicious modes will be discussed later in this paper. Under the confident mode, he is confident about himself and continues its current plan. Otherwise, it tends to be cautious, and reconsiders its intentions every time before performing any action. The human continually executes a cycle of observing the environment, deciding what intention to achieve next, determining a plan to achieve this intention, and then executing this plan. The details of the control loop algorithm (pseudo code), corresponding to the confidence mode, are shown in Figure 4 [Wooldridge, 2000]. This control loop algorithm is referred to as single-minded commitment strategy of the human (see Section 3.4.3). The open-minded commitment strategy, corresponding to the suspicious mode, will be discussed in Section 3.4.3.
### Human Operator Agent Control Loop Algorithm

1. $B := B_0$ ;  // $B_0$ are initial belief set
2. $I := I_0$ ;  // $I_0$ are initial Intentions
3. while true do
4.   get next percept $\rho$ ;  // Get new percept $\rho$
5.   $B := \text{brf}(B, \rho)$ ;  // Based on new percept update the belief set
6.   $D := \text{options}(B, I)$ ;  // Based on current belief and intention to update desire set
7.   $I := \text{filter}(B, D, I)$ ;  // Select a desire to commit to, and it becomes intention.
8.   $\Pi := \text{plan}(B, I)$ ;  // Generate alternative plans
9.   $\pi := \text{decision}(\Pi)$ ;  // Select one plan to execute
10. while not (empty ($\pi$) or succeeded($I, B$) or impossible($I, B$) ) do
11.   $\alpha := \text{hd}(\pi)$ ;  // $\text{hd}(\pi)$ is the first action in the plan $\pi$
12.   execute $\alpha$ ;
13.   $\pi := \text{tail}(\pi)$ ;  // $\text{tail}(\pi)$ is all the actions except the first action in the plan $\pi$
14.   get next percept $\rho$ ;
15.   $B := \text{brf}(B, \rho)$ ;
16.   if not sound($\pi, I, B$) then
17.       $\Pi := \text{plan}(B, I)$
18.   end-if
19. end-while
20. if impossible ($I, B$), post the problem to others
21. end-if
22. end-while

Figure 4. Human’s control loop algorithm (single-minded commitment strategy)

### 3.3 Mental State Models of Human Operator

In the proposed model, the human is completely specified based on the events that it can perceive, the actions it may perform, the beliefs and confidence index it may hold, the desires it may adopt, and the plans that give rise to its intentions [Kinny et al., 1996]. These are captured by the beliefs in the belief module, desires in the desire module, and intentions in the decision-making module. In this work, a modified LORA logic is employed to represent the beliefs, desires, and intentions. The LORA logic was chosen because of its rich expressiveness, combining first-order logic, temporal logic, and modal logic. Table 1 depicts an overview of the exemplary belief, desire, and intention in LORA form [Wooldrige, 2000].
Table 1. LORA Belief, Desire, Intention modalities

<table>
<thead>
<tr>
<th>Formula</th>
<th>Interpretation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Bel $i \varphi$)</td>
<td>agent $i$ believes $\varphi$, where $i$ is an agent’s name, $\varphi$ is any formula of LORA.</td>
<td>(Bel $operator\ Robot(\text{free})$), the operator agent believes that the robot is free.</td>
</tr>
<tr>
<td>(Des $i \varphi$)</td>
<td>agent $i$ desires $\varphi$</td>
<td>(Des $operator\ part1(\text{buffer2,machine})$, The agent $i$’s desired state is that part 1 finished its operation on machine and in buffer 2.</td>
</tr>
<tr>
<td>(Int $i \varphi$)</td>
<td>agent $i$ intends $\varphi$</td>
<td>(Int $operator\ part1(\text{buffer1,\neg machine})$, The agent $i$’s intended state is that part 1 did not finished its operation on machine and in buffer 1.</td>
</tr>
</tbody>
</table>

3.3.1 Belief Model

A belief model (or beliefs) describes information about the environment, internal states of a human, and the actions that he/she can apply to change the states of the environment. The beliefs of an operator are denoted by a 3-tuple: $<\text{Environment, Actions, Itself}>$. The structure of the beliefs used in this paper is shown in Figure 5, and the details are discussed in the following subsections.

![Beliefs of Human Operator](image)

Figure 5. The hierarchical structure of human’s beliefs

1) Beliefs about Environment

The environment for the case of the automated manufacturing system is characterized by the pieces of equipment pertaining to the system and the parts processed in the system (see Figure 5). In this paper, we focus only on the equipment level states of the system, but will not
consider the states of the accessories (e.g. tools, pallets and sensors). In addition, only dynamic states that change during the system operation are of our interest, while constant values (e.g. relative positions among the equipment) are not considered.

**State of Equipment:** Several classes of equipment have been identified in a manufacturing system. Smith and Joshi [1993] have defined these classes of Equipment \( E \) as Material Processing (MP), Material Handling (MH), Material transporting (MT), and Buffer Storage (BS), and can be formally defined as \( E = <MP, MH, MT, BS> \). The possible states of the equipment include busy, idle or breakdown, and are formally represented using the following LORA predicates and composite predicates:

\[
\begin{align*}
\text{equipment (busy)}, \text{equipment(idle)}, \text{equipment(breakdown)} \\
\text{equipment (busy)} \land \text{equipment(breakdown)}, \text{equipment(idle)} \land \text{equipment(breakdown)}
\end{align*}
\]

**State of Part:** As shown in Figure 5, the state of a part is defined by its current status and its process plan. Its current status is composed of its position and its finishing status of each required operation. Regarding the position, only addressable positions are defined. By addressable position we mean a part is in equipment or predefined positions. For example a part is in a machine or in a predefined position. An exemplary, current status of a part is shown below, where two predicates denote that the part is in equipment/position now, it already finished its operation on equipment \( mp1 \), and does not finish its operation on equipment \( mp2 \) yet.

\[
\begin{align*}
\text{part(equipment, } mp1, ¬mp2 ,...) \\
\text{part(position, } mp1, ¬mp2 ,...)
\end{align*}
\]

As another example, \( part1(robot, m1, ¬m2) \) denotes that \( part1 \) is now in a robot, it has finished its operation on equipment \( m1 \), but has not finished on equipment \( m2 \) yet.

The process plan is associated with actions, and will be discussed after discussion of the actions (see Section 3.3.1.2.b).

**Properties:** There are relationships among the states of the equipment and parts. For example, the following properties are always true:

\[
\begin{align*}
\forall x, y, x \in \text{part}, y \in \text{equipment}, x(y, ...) \Rightarrow ¬y(idle) \\
\forall y, y \in \text{equipment, } y(busy) \Rightarrow \exists x, x \in \text{part, } x(y,...) \\
\forall y, y \in \text{equipment, } y(busy) \Leftrightarrow ¬y(idle) \\
\forall y, y \in \text{equipment, } y(idle) \Leftrightarrow ¬y(busy)
\end{align*}
\]
The first property denotes that if a part $x$ is in equipment $y$ then equipment $y$ must not be in an *idle* state. The second property denotes that if equipment $y$ is *busy* then there must exist a part $x$ which is in an equipment $y$. The third and forth properties denote that the *busy* and *idle* states of equipment are exclusively true.

2) **Beliefs about Actions**

*Actions*: Equipment executes an *action* to change the states of a manufacturing system. As shown in Figure 5, the actions are partitioned to *high level actions* and *basic actions*. The basic action is a typical single action of equipment in the manufacturing system. The high level actions are composite actions or plans, which can be obtained from the previous, successful experiences. Through the composite actions, we intend to enable the learning ability of the human model. The learning ability, however, is beyond the topic of this paper. The high level actions can also be the actions or plans of another system that the human has inherited. For example, the human directly inherits the actions or plans of the automated system and stores it in the high level action set. In the human’s planning process, the high level actions are selected first if available. Otherwise, the basic actions are selected. This policy (higher preference to the high level actions) facilitates the monitoring function of the human agent, which will be discussed later in this paper (see Section 5).

In this work, the STRIP style structure [Fikes and Nilsson, 1971] is employed to represent the actions. Each action consists of three STRIPS-style operator elements. First, the ADD list contains new predicates that the action causes to become true. Second, the DELETE list contains old predicates that the action causes to become false. Finally, the PRECONDITION list contains those predicates that must be true for the action to be applied. Table 2 depicts the typical actions of material processing (MP), material handling (MH), and material transporting (MT) equipment.

*Process Plan*: The process plan is best illustrated by an example. Suppose we have a process plan $p$ for part1 as shown in Figure 6, in which the arcs are labeled as actions. Then, the process plan can be represented as:

\[
\begin{align*}
&((\text{Happens }a1;(a2\mid a3))
\end{align*}
\]

\[
\begin{align*}
&((\text{Happens } (m2\mid a3)|(m3\mid a2))
\end{align*}
\]
### Table 2. Typical actions of MP and MH/MT

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Typical actions</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MP</strong></td>
<td><strong>PROCESS((part, mp))</strong></td>
<td>Process part in equipment mp. The preconditions: part has not processed on mp yet and part is in mp. The add list: part has processed on mp and the part is in mp. The delete list: part has not processed on mp yet and part is in mp.</td>
</tr>
<tr>
<td></td>
<td>Preconditions: (part(mp, ¬mp))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add list: (part(mp, mp))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete list: (part(mp, ¬mp))</td>
<td></td>
</tr>
<tr>
<td><strong>MH/MT</strong></td>
<td><strong>MOVE((part, equipment))</strong></td>
<td>Move part to equipment</td>
</tr>
<tr>
<td></td>
<td>Preconditions: (part(robot, ...) \land equipment(\text{free}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add: (part(equipment, ... \land equipment(busy)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete: (part(robot, ... \land equipment(\text{free}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PUT((part, equipment))</strong></td>
<td>Put part to equipment</td>
</tr>
<tr>
<td></td>
<td>Preconditions: (part(robot, ... \land equipment(\text{free}) \land robot(busy)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add: (part(equipment, ... \land equipment(busy) \land robot(\text{free}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete: (part(robot, ... \land equipment(\text{free}) \land robot(busy)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PICK((part, equipment))</strong></td>
<td>Pick part from equipment</td>
</tr>
<tr>
<td></td>
<td>Preconditions: (part(equipment, ... \land equipment(busy) \land robot(\text{free}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add: (part(robot, ... \land equipment(\text{free}) \land robot(busy)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete: (part(equipment, ... \land equipment(busy) \land robot(\text{free}))</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6. A process plan example**

```
P
Part(x, ¬m1, ¬m2, ¬m3)
  \downarrow a1: PROCESS(part1, m1)
Part(x, m1, ¬m2, ¬m3)
  \downarrow a2: PROCESS(part1, m2)
  \downarrow a3: PROCESS(part1, m3)
Part(x, m1, m2, ¬m3)
  \downarrow a3: PROCESS(part1, m3)
Part(x, m1, m2, m3)
  \downarrow a2: PROCESS(part1, m2)
Part(x, m1, m2, m3)
```
“Happens” is a formal LORA operator used to represent actions. The first LORA expression denotes that action $a1$ happens first followed by $a2$ or $a3$. The second expression denotes if the operation on $m2$ is finished first, then the next action is $a3$. Or else if the operation on $m3$ is finished first, then the next action is $a2$. This example reveals that the LORA is capable of representing alternative process plans, which are essential elements in real-time decision-making. Table 3 depicts typical constructors for the action expression [Wooldridge, 2000].

Table 3. Typical constructors for action expression

<table>
<thead>
<tr>
<th>Expression</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha;\alpha'$</td>
<td>$\alpha$ followed by $\alpha'$</td>
</tr>
<tr>
<td>$\alpha \mid \alpha'$</td>
<td>either $\alpha$ or $\alpha'$</td>
</tr>
<tr>
<td>$\varphi?$</td>
<td>$\varphi$ is satisfied</td>
</tr>
</tbody>
</table>

3) **Beliefs about Agent Itself**

The human’s belief about himself/herself is represented by the confidence level about his/her performance. The confidence level is represented by its Confidence Index (CI). The CI is calculated using the following equation, where $j$ is the total number of actions that the human executed until current time, $\alpha_i$ is the weight of $i$th action, $K$ is an integer constant, $b$ is a constant ($0 < b < 1$), $\text{CI}_0$ is the initial confidence index (constant).

$$CI = \text{CI}_0 + \sum_{i=0}^{j-1} \alpha_i,$$

$$i_0 = \begin{cases} 0, & \text{if } j - K < 0 \\ j - K, & \text{if } j - K \geq 0 \end{cases},$$

$$\alpha_i = \begin{cases} -b^{(j-i)}, & \text{when an action fails} \\ b^{(j-i)}, & \text{when an action succeeds} \end{cases},$$

where:

Based on this equation, the operator evaluates the successfulness of its actions and updates its confidence level. The confidence mode (two discrete modes are currently considered in this paper) of the human is a function of the confidence index (see below).

$$\begin{cases} CI \geq T_{ci}, \text{The operator is in happy state} \\ CI < T_{ci}, \text{The operator is in sad state} \end{cases},$$
$T_{CI}$ is a predefined threshold value for confidence index $CI$. As discussed before, the agent’s confidence index determines the agent’s decision-making strategies, which will be discussed in Section 3.4.3.

### 3.3.2 Desire and Intention Model

A desire model (or desires) constitutes the goal-oriented analogue of an activity specification, consisting of a hierarchical collection of goals. The desire state is denoted by a 2-tuple: $<\text{Scheduling level, Equipment level}>$. In this paper, our focus is on the execution level of the manufacturing system, therefore scheduling level desires and equipment level desires are considered. For other applications, planning or/and tool levels desires can also be considered. The desires are partitioned to long-run desires and current desires. Current desires are the goal states which the agent is currently trying to be brought about. As the operation proceeds, the long-run desires become current desires. The conflicts in the equipment level current desires can be solved based on the scheduling level current desires and long-run desires. The desire model can be represented as a hierarchical structure (see Figure 7) similar to the belief model.

The intentions can be represented in same manner as desires; it is noted that intentions are subset of desires. Desires and Intentions are in the form of the goal states that the agent intends to be brought about. The symbol “◊” means “sometime”, thus “◊φ” means φ is eventually true. An exemplary desires and intentions in LORA logic are shown in Table 4.

![Figure 7. The hierarchical structure of operator’s desires](image)

**Table 4. An operator’s desires and intention example**

<table>
<thead>
<tr>
<th></th>
<th>Scheduling level</th>
<th>Execution level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desires</td>
<td>$◊ ((\text{part1}(\text{buffer3}, m1)) \land \text{part2}(\text{buffer4}, m1))$</td>
<td>$◊ ((\text{part1}(\text{buffer3}, m1))$</td>
</tr>
<tr>
<td>Intentions</td>
<td>$◊ ((\text{part1}(\text{buffer3}, m1))$</td>
<td>$◊ ((\text{part1}(\text{robot, ¬m1}))$</td>
</tr>
</tbody>
</table>
3.4 Planning Process and Corresponding Dynamics of Mental States

In this section, an exemplary scenario is used to illustrate the planning process and its corresponding dynamics of the agent’s mental states. The exemplary scenario is that a part, being in a buffer now, needs to be processed on equipment $m$ and put back to the buffer.

- **Initial beliefs** about the environment:
  
  \[
  \text{Bel operator robot(} \text{free}) \land m(\text{free}) \land \text{buffer(busy)}
  \]
  
  \[
  \text{Bel operator part(buffer,} \neg m) \land (\text{Happens PROCESS}(part,m))
  \]

- **Initial beliefs** about itself
  
  Bel_I operator CI = 0

- **Initial beliefs** about actions

  \[PICK(part, equipment)\]
  
  Precondition list: \(part(equipment) \land \text{robot(} \neg \text{free})\)
  
  Add list: \(\text{robot(busy)} \land \text{part(robot)}\)
  
  Delete list: \(\text{robot(} \text{free}) \land \text{part(equipment)}\)

  \[PUT(part, equipment)\]
  
  Precondition list: \(part(\text{robot}) \land \text{equipment(} \neg \text{free})\)
  
  Add list: \(\text{robot(} \text{free}) \land \text{equipment(busy)}\)
  
  Delete list: \(\text{robot(busy)} \land \text{equipment(} \text{free}) \land \text{part(robot)}\)

- **Initial desires and intentions**
  
  Des operator $\Diamond part(buffer, m)$
  
  Int operator $\Diamond part(buffer, m)$

3.4.1 Planning and Decision-Execution

Given the current beliefs and intentions of the agent, the agent first generates alternative plans (planning) to achieve its intentions. The signature of the planning is presented below.

\[\text{plan : } \wp(\text{Bel}) \times \wp(\text{Int}) \rightarrow \text{Plan}\]

The planning process is responsible for the reasoning ability of the human before he/she makes a decision regarding actions. By applying actions, the agent generates plans that will lead it from his/her current state to the goal state. In this work, STRIPS-style planning is employed to represent the agent’s planning process. Figure 8 depicts an instance of the real-time planning procedure. In step 1, the operator compares the current beliefs (current state) with the intentions
(goal state). If the current beliefs confirm with the intentions, the intentions have been achieved and the planning process stops. Otherwise, the agent applies one of the actions such that its preconditions are satisfied by its current beliefs (Step 2). After a specific action is applied, the current beliefs are updated. The predicates in the DELETE list are deleted from the beliefs, and the predicates in the ADD list are added to the beliefs (Step 3). Here, the agent plans for the future state changes in a logic manner. However, the plan may or may not be true in the real system because of possible unanticipated errors. This is different from the belief revision function, which is responsible for the real-time belief updating (see Section 3.4.2). The planning procedure continues until the intentions (goal state) are reached. The obtained alternative plans will be stored in a plan set (plans). Finally, the agent selects an optimal/satisfactory plan based on decision-models and executes it via the decision executor.

Figure 8. Real-time planning procedure

3.4.2 Belief Revision Function (brf)

The beliefs can be updated through the new percepts forwarded by the sensor in the environment. The brf can be formally represented as \( brf : \varphi(Bel) \times Per \rightarrow \varphi(Bel) \). In other
words, on the basis of the current beliefs and current percept, the belief revision function determines a new set of beliefs. The agent’s beliefs about himself/herself are also updated as shown in Section 3.1.1.3. The actions also can be updated. If a plan is executed successfully, the plan is stored in the high level actions as a composite action (see Section 3.3.1.2).

3.4.3 Agent’s Confidence Index and Commitment Strategies

Based on the confidence index, the agent selects different commitment strategies. The confident mode is taken to be the mood of achieving the goals, and being engaged in what it is doing. It is triggered when the agent performed tasks successfully [Meyer, 2004]. The associated commitment strategy is referred to as single-minded strategy (see Figure 4). The characteristic of this strategy is that the agent continues to maintain an intention until it believes that either the intention has been achieved, or it is no longer possible to achieve the intention [Wooldridge, 2000].

The suspicious mood is the emotion of losing goal, and knowing it cannot be reinstated. It is triggered by the failure of a major plan or a continued sequence of actions [Meyer, 2004]. The associated commitment strategy is referred to as open-minded strategy (see Figure 9). The characteristic of this strategy is that the agent is cautious and reconsiders intentions every time before performing an action (see steps 16 and 17 in Figure 9).

<table>
<thead>
<tr>
<th>Human Operator Agent Control Loop Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $B := B_0$;</td>
</tr>
<tr>
<td>2. $I := I_0$;</td>
</tr>
<tr>
<td>3. while true do</td>
</tr>
<tr>
<td>4. get next percept $\rho$;</td>
</tr>
<tr>
<td>5. $B := \text{brf}(B, \rho)$;</td>
</tr>
<tr>
<td>6. $D := \text{options}(B, I)$;</td>
</tr>
<tr>
<td>7. $I := \text{filter}(B, D, I)$;</td>
</tr>
<tr>
<td>8. $\Pi := \text{plan}(B, I)$;</td>
</tr>
<tr>
<td>9. $\pi := \text{decision}(\Pi)$;</td>
</tr>
<tr>
<td>10. while not (empty ($\pi$) or succeeded($I, B$) or impossible($I, B$)) do</td>
</tr>
<tr>
<td>11. $\alpha := \text{hd}(\pi)$;</td>
</tr>
<tr>
<td>12. execute $\alpha$;</td>
</tr>
<tr>
<td>13. $\pi := \text{tail}(\pi)$;</td>
</tr>
<tr>
<td>14. get next percept $\rho$;</td>
</tr>
</tbody>
</table>
15. \( B := \text{brf}(B, \rho); \)
16. \( D := \text{options}(B, I); \)
17. \( I := \text{filter}(B, D, I); \)
18. if not sound(\( \pi \), I, B) then
19. \( \Pi := \text{plan}(B, I); \)
20. end-if
21. end-while
22. if impossible (I, B), post the problem to others
23. end-if
24. end-while

Figure 9. Open-minded commitment strategy

4. Integration of Operator Model with Automated Control System

In this section, the scheme of integrating of the proposed agent model with an automated shop floor control system is presented. Our focus is on the real-time system monitoring and error recovering responsibilities of the human agent. Monitoring a complex automated system needs continuously attentions and could be quite boring. Therefore, the real human operator can easily get tired from this task and lose the track of the actions of the automated system. As a result, when accident occurs the operator doesn’t have enough information of the system and fails to respond to it timely and correctly. Furthermore, monitoring procedures are rote steps, and can be therefore easily coded into the proposed agent model. Due to the dynamics and complexity of the manufacturing system, unexpected circumstance is an every day occurrence in the automated system [Grier and Hsiao, 1990]. Therefore, the real-time error recovery is essential to the success of the automated system. The reasoning ability of the proposed agent model enables the real-time diagnosis and generation of a plan used to recover the system.

In the automated shop floor control system (see Figure 10), the ERP system sends orders to the Scheduling Level Task Generator (SLTG), which then generates the sequences of tasks required to produce the orders. The task generator sends the tasks to the Execution Level Action Generator (ELAG), and which then generates sequences of actions to implement the task. The actions are sent to the Manufacturing Execution System (MES) to execute them through equipment controllers in the physical shop floor. When an action/task is completed, an ‘ok’
message is sent to ELAG/SLTG and the system continues. The control loop of the automated system is described in Figure 11.

Figure 10. An automated shop floor control system

1. based on ERP information, the SLTG (Scheduling level Task Generator) generates the sequence of tasks
2. input the first task into GLAG (Execution Level Action Generator)
3. determine corresponding sequence of actions to implement the task (execution level)
4. input the first action in to physical controllers
5. if preconditions of the first action evaluate correctly
6. if the action executes correctly
   \{change states and remove the action from the actions list; if the task is the last action of a task, remove the task from the task list too\}
7. end-if
8. end-if

Figure 11. Control loop of a typical automated shop floor control system

This automated system works well under the predefined control routines, but it fails under unexpected circumstances. For example, when the preconditions of an action are not met or the action itself fails, the system is most probably hang. One may argue that we can add more
control logic in this system to avoid the problem. For example, when the preconditions of an action are not met, then the system can go to another predefined subroutine which is able to solve this precondition error. However, the problem is that it is impracticable or uneconomical to design control logic for all possible precondition errors in advance, since the combinations of the system states can be very large. For example, in a system with m resources and n parts, if we assume each resource and part has a and b numbers of status, the combinations of them results in $a^m \times b^n$. This number can be very large even in a simple, realistic manufacturing system. On the other hand, the number of the generic action types that can be applied to change the states of the system is quite small. Furthermore, parts are passive and their states can only be changed through the actions of equipment. For example, a major action of the material processing equipment is to process parts. This action can be defined as a generic $\text{PROCESS}(\text{part}, \text{equipment})$ action, and the different parameters ($\text{part}$, and $\text{equipment}$) can be specified to indicate different processing programs. However, the effects of $\text{PROCESS}(\text{part}, \text{equipment})$ with different parameters are the same, which changes the states of the $\text{part}$ from not being processed on equipment to already being processed on the equipment. There is only one generic action type for the material processing equipment. Even when its tool level accessories are considered, there are still only few generic actions. Therefore, some limited number of generic actions can be searched (reasoned) in the system state space, which may contain a very large number of system states.

According to the control procedure in Figure 11, two types of generic errors may arise in the automated system: precondition error and action error [Wu and Joshi, 1994]. Exemplary precondition errors include 1) program error, 2) communication error, 3) timing error, and 4) positioning error. Exemplary action errors include 1) machine failure, 2) collision, 3) operation error, and 4) dropping part.

The architecture of the integrated shop floor control system (integrating the automated control system and the proposed agent) is depicted in Figure 12. There are three major components in the architecture, including 1) human agent model, 2) automated shop floor control system, and 3) the physical shop floor.
Figure 12. Integrated BDI human model with automated system

To solve the previous mentioned problems of the automated control system, we propose to integrate it with the proposed agent model. In the proposed integrated system, the SLTG sends a task command to both the ELAG and the Cognitive Processor of the agent model (denoted by line A in Figure 12). The ELAG also sends an action to both MES and the Cognitive Processor of the agent model (denoted by line B in Figure 12). After the Cognitive Processor receives the current first task and first action, it generates the appropriate two-level current Desires (current scheduling level Desires are the desired goal states after the current task is executed, and current execution level Desires are the desired goal states after the current action is executed). The agent model is designed to commit to all the Desires inherited from the automated system, and therefore these Desires automatically become the Intentions of the agent. As mentioned before (Section 3.2), the operator directly inherits the automated system’s intentions. The ELAG also sends the current action to the belief module (denoted by line C in Figure 12), and which is then stored in high level action set. In the planning stage, the preconditions of the action will be compared with the current Beliefs (precondition error
If the action is acceptable, the Planning process of the agent comes up with the same plan (only one action in our case) as the automated system. This is because the ELAG’s action has been already stored in the high level action set, and the high level action is selected first. The agent then sends it to the MES (line D in Figure 12) and the action is executed through the physical shop floor controllers (monitoring mode). If the action is not acceptable, it implies a precondition error occurred. The Planning will then generate new plans other than the one inherited from the automated system to achieve the Intentions (precondition error recovery).

After an action is executed, the Beliefs are updated based on the information from the sensors in the physical shop floor. If the current goal state is met (monitoring mode), then 1) the action is deleted, 2) the agent sends an “ok” message to the MES, 3) the “ok” message is sent back to corresponding ELAG/SLTG of the automated control system and the system continues. If the goal state does not confirm with the current Beliefs (action error diagnosis), it implies the action has failed. The operator will generate new plans to reach the goal state based on the current Intentions and the updated Beliefs (action error recovery). Since some critical action errors, such as collision, tool breakage, and dropping part, may cause further damage, they must be handled in a timely manner. To handle these errors, the corresponding remedial plans can be developed in advance. Whenever this kind of error is detected, the agent directly executes the corresponding remedial plans first. This is implemented through the event-driven reactive action defined in the agent model, and a relevant example is shown in Section 5.1. The control loop of the integrated system is shown in Figure 13.

```
1. based on ERP information, the SLTG (Scheduling level Task Generator) generates the sequence of tasks
2. input the first task into GLAG (Execution Level Action Generator)
3. determine corresponding sequence of actions to implement the task (execution level)
4. update the Intentions, update high level actions Beliefs
5. go to the BDI operator control loop
6. execute the action
7. update the Beliefs based on new perception
8. if the current intention is met
   {change states and remove the action from the actions list; if the task is the last action of a
```
Figure 13. Control loop of the system integrating the automated control system and agent

5. Illustration with Example and Implementation Issues

The JACK software is used to model the proposed human agent. JACK is an agent oriented development environment built on top of and fully integrated with the JAVA programming language. A JACK agent is a software component that can exhibit reasoning behavior under both the pro-active (goal-oriented) and reactive (event-driven) stimuli. JACK provides software modules to construct beliefs, desires, intentions, events, and plans. However, JACK does not support real-time generation of a plan (it only supports static/predefined plans). Currently, active research efforts are ongoing to implement the real-time reasoning mechanism in JACK software. In this paper, the planner sub-module has been developed in Java, which is callable from JACK software. The planner sub-module generates the appropriate plans for different intentions.

Two types of errors (collision and deadlock) are considered in this section to illustrate and test the proposed human agent model. We assume that the system consists of two material processing equipment (m1 and m2), a robot and four buffers (b1, b2, b3, and b4) (see Figure 14). It is also assumed that the sequence of tasks generated from the task generator is: “process part1” and “process part2”.

Figure 14. Exemplary shop floor
5.1 Collision (Event-Driven Reactive Action)

It is assumed that when the robot executes the action $PUT(part1,m1)$, a collision occurs (the torque of the robot joint exceeds the threshold value). To avoid further damage, the robot stops and waits for the agent to recover from the error. Usually, action plans are predefined to respond to this kind of critical errors in a timely manner. It is supposed that the collision occurs when the robot becomes inaccurate, and the robot regains its accurate after it is recalibrated (through $HOME(robot)$ action). The current beliefs and predefined plan for the robot collision are shown below.

- The current beliefs of the environment:
  
  $Bel\ operator\ Robot(collision) \land M1(free) \land M2(free) \land buffer1(free)$
  
  $Bel\ operator\ Part1(robot,\neg m1) \land (Happens\ process(\ part1,\ m1))$

- The predefined plan for robot collision (CollisionRecovery):
  
  $(Happens\ MOVE(\ robot, s1);\ HOME(\ robot))$

This event-driven reactive action is modeled in JDE Graphical Plan Editor of JACK as shown in Figure 15. When the sensors detect the collision, the collision event is sent to the operator and the operator the uses the predefined plan (CollisionRecovery, move the robot to a predefined safe place ($s1$) and home the robot) to handle it.

![Figure 15. Event-driven collision handling](image)

5.2 Deadlock (Goal-Oriented Proactive Action)

One of complex problems in the BDI agent is the intention confliction problem. In the proposed agent model, the intentions are presented in a hierarchical structure. We propose to use
the higher level intention as a guideline to resolve the lower level intention confliction. For example, we assume part1 just finished its operation in m1 and part2 are being processed in m2. When the robot picks up part3 and intends to put it to m1, a deadlock occurs in the system. If we want to put part 3 to m1, m1 must be freed first. To free m1, the robot needs to pick part1 first, but the robot is already busy (holding part3).

- Current beliefs of the environment:
  Bel \(\text{operator} \ Robot(busy) \land M1(busy) \land M2(busy) \land \text{buffer1}(\text{free})\)
  Bel \(\text{operator} \ Part1(m1, \neg m1) \land (\text{Happens process}(part1, m1))\)
  Bel \(\text{operator} \ Part2(m2, \neg m1) \land (\text{Happens process}(part2, m2))\)
  Bel \(\text{operator} \ Part3(\text{robot}, \neg m1) \land (\text{Happens process}(part3, m1))\)

- Intentions:
  Int \(\diamond \text{operator} \ Part1(\text{buffer3}, m) \) (scheduling level)
  Int \(\diamond \text{operator} \ Part3(m1, \neg m1) \) (execution level)
  Int \(\diamond \text{operator} \ Part3(m1, \neg m1) \) (current intention)

Based on current beliefs and intention, the deadlock error can be resolved by using the scheduling level intention as a guideline. Since the scheduling level intention intends to finish part1 first and put it to buffer 3, the robot needs to be freed first to perform picking operation of part 1 from m1. Using the planning procedure, we can come up with a plan as shown below:

\((\text{Happens PUT}(m3, \text{buffer1}); \text{PICK}(part1, m1); \text{PUT}(part1, b3))\).

To handle the precondition error, the goal-oriented planning discussed in Section 3.4.1 is coded in Java and integrated into JACK.

6. Distributed Computing Platform

In this work, we have developed a distributed computing platform based on DOD High Level Architecture (HLA) (now IEEE 1516 Standard) to facilitate integration of an agent (JACK software, Java application in charge of planner), real human, and the environment (real or simulated (Arena software)). Figure 16 illustrates the relationships between the components of the agent and real-human in-the-loop system. The agent can be implemented in different languages (e.g., JACK, AgentSpeak, Jadex, etc). The agent can interact with either a real environment (automated shop floor control system in our case) or a simulated environment.
Also, the simulator (environment) can be implemented in different languages (e.g., Arena™, AutoMod™, ProModel™, etc). The direct interaction of the application software (both agent software as well as simulation software) with the RTI (executable component of HLA) is quite complex and cumbersome. Therefore, we have developed a Distributed Simulation (DS) Adapter, based on COM/DCOM technology, to provide mechanisms for distributed application similar to those provided by the HLA RTI, but with a level of complexity that is manageable by the development resources available in the non-computer science communities. The DS Adapter provides a simplified time management interface, automatic storage for local object instances, management of lists of remote object instances of interest, management and logging for interactions of interest, and simplified object and interaction filtering.

Figure 16: Integration of agent, human, and environment (real or simulated) based on HLA and DS Adapter

7. Contributions and Conclusions

This paper presented a novel software agent model to replace the partial decision-making function of a human. The proposed agent model impacts several different areas, including Artificial Intelligence (AI), manufacturing, and distributed simulation areas, as well as Air Force applications. The AI contributions can be summarized as follows. First, we have developed a human decision-making model, which is capable of 1) generating a plan (a sequence of actions required to achieve a goal) in real-time as opposed to selecting a plan based on a static algorithm and predefined/static plan templates that have been generated off-line, 2) supporting both the
reactive as well as proactive decision-making, and 3) maintaining situation awareness in human language like logic to facilitate real human decision-making. Second, the proposed human decision-making model enhances the traditional BDI framework, expanding the intention module to include 1) deliberator sub-module, 2) planner sub-module, and 3) decision executor sub-module and considering an emotion state, which affects as well as is affected by the belief module, desire module, and decision-making module.

To the best of our knowledge, the proposed model is the first human operator decision-making model in the manufacturing area capable of human like goal-oriented reasoning. Also, the contribution of this work to the field of automated shop floor control is significant, allowing seamless integration among the control system, an agent, and real-human.

The Distributed Simulation (DS) Adapter (based on COM/DCOM technology) that we have developed in this work provides mechanisms for distributed application similar to those provided by the HLA RTI, but with a level of complexity that is manageable by the development resources available in the non-computer science communities.

The proposed human decision-making model is targeted for a complex system, where a human must adaptively adjust his/her behavior to the dynamically changing environment. Although our work has been developed and demonstrated in the context of the error detection and recovery personnel in a complex automated manufacturing environment, it is expected that the model is directly applicable to the human operators dealing with complex systems in Air Force (e.g. pilots) and in civilian systems such as operators in a nuclear reactor, power plant, and extended manufacturing enterprise.

References


