MULTI-AGENT SYSTEMS FOR ENGINEERING DESIGN OF FIXTURES

[Biography]

Dr. Joshua D. Summers received his doctoral degree in Mechanical and Aerospace Engineering from Arizona State University in 2004. He is currently Associate Professor of Mechanical Engineering at Clemson University and directs the Automation in Design (AID) research group. His primary research interest is centered on developing design enabling tools for industry practice through studying, practicing, and teaching engineering design.

Dr. Farhad Ameri received his doctoral degree in Manufacturing Engineering from the University of Michigan, Ann Arbor, in December 2006. He is currently a postdoctoral research fellow in the Automation in Design (AID) research group in the Mechanical Engineering Department at Clemson University. His primary research interests include distributed design and manufacturing, engineering information systems, and product lifecycle management.
MULTI-AGENT SYSTEMS FOR ENGINEERING DESIGN OF FIXTURES

FARHAD AMERI, JOSHUA D. SUMMERS

Automation in Design Research Group
Department of Mechanical Engineering
Clemson University
CLEMSON, SC 29634-0921, USA

Abstract. This chapter introduces an agent-based framework for engineering design of fixture systems. A number of researchers have applied the agent technology to engineering design and several agent-based systems have been developed accordingly addressing different application domains in design. Development of multi-agent system for fixture design is motivated by the need for a computational infrastructure that supports integration of disparate Computer-aided Fixture Design (CAFD) systems in distributed environments. In this chapter, multi-agent approach to fixture design is studied from two perspectives: 1) Agent Architecture and 2) System Architecture. At an agent-level, a problem solving methodology for fixture design based on evolutionary algorithms is discussed. At a system-level, a generic framework for agent interaction and communication for problem definition, fixture synthesis, and fixture analysis is proposed in this chapter. The authors hope this chapter can help researchers gain a better understanding of the major challenges, research issues and future directions in the domain of agent-based fixture design.

Keywords: Intelligent Engineering Design, Agent Based Design, Information Modeling, Fixture Design, Design Automation

INTRODUCTION

Today, due in part to the globalization of the economy, engineering design practice has increasingly become distributed and collaborative. In a distributed environment, some of the challenges associated with collocated design practice, such as management of engineering information and knowledge, communication and collaboration among different disciplines, and integration of design automation tools are intensified (Chira, 2006). Traditional approaches such as legacy system integration or establishment of data exchange standards cannot sufficiently address the emerging needs of today’s collaborative design environments because of the highly distributed nature of design teams, diversity of engineering tools and complexity and dynamics of design environments (Wang, 2002). Distributed design process should be supported by intelligent computational technologies that enable collaborative problem solving by dispersed and autonomous decision makers in heterogeneous environments. One such technology is Multi-agent systems (MAS) which represent a potential solution to the distributed design problem.
In the domain of Artificial Intelligence, an agent is a software program that is capable of independent action on behalf of its user or owner. A multi-agent system consists of several agents that interact with one another. Several ongoing trends have contributed to the emergence of agent-based systems in the field of computer science. These trends include ubiquity, interconnection, intelligence, delegation and human-orientation (Woodridge, 2002). Ubiquity refers to the continual reduction in the cost of computing that has resulted in economic impartation of processing power into distributed devices. Interconnections points to the networked nature of computer systems that has necessitated continual interaction of dispersed software entities. Steady growth of intelligence, or complexity of processing tasks performed by computers, is the next influential trend. Increased delegation of tasks to computers is another visible trend that has led to the development of more sophisticated computer systems such as MAS. The last but not the least is human-orientation which can be interpreted as departing from machine-oriented views of programming to the approaches that more closely resemble human way of perception, learning, and action.

Agents are best suited for applications that are modular, decentralized, changeable, ill-structured, and complex (Parunak, 1999). Distributed product development is one such application. A number of researchers have applied the agent technology to engineering design and several agent-based systems have been developed accordingly such as PACT (Cutkosky et al., 1993), DIDE (Shen, 1995), SHARE (Toye, 1994), and First-Link (Park, 1994) addressing different application domains in design. This chapter provides an overview of existing works in agent-based engineering design and, in particular, focuses on application of agent-based systems in fixture design. In order to provide the readers with a clearer picture of the concepts and methods described in this chapter, an illustrative framework for agent-based fixture design is presented toward the end of this chapter which is referred to as AFFIXED (Agents For FIXturE Design) throughout this chapter.

Extensive use of fixtures with varying complexities in almost every phase of manufacturing necessitates systematic approach to fixture design process. Several Computer-Aided Fixture Design (CAFD) systems have been introduced to the market in order to automate the fixture design task. However, existing CAFD systems are not built at the outset for distributed and collaborative environments. To address the needs of distributed and collaborative design environments, next-generation fixture design solutions should satisfy several fundamental requirements including possibility of integration with different engineering design tools, ability to operate in heterogeneous environments, scalability based on the size and complexity of fixture design problem, flexibility to accommodate different categories of design problems, and interoperability to enable seamless information exchange among heterogeneous software components. Also, given the fact that fixturing elements are increasingly becoming standard within organizations that have replicated production environments throughout the world, a distributed computational framework can significantly enhance reuse of fixture design artifact and fixture design knowledge. A multi-agent system, by its very nature, can meet the aforementioned requirements. Furthermore, given the multiplicity of possible solutions for a particular fixture design problem as well as distribution of knowledge in the open, decentralized and, dynamic design environment envisioned for fixture development, fixture design problem lends itself well to agent-based scenarios. It should be emphasized that, based on the lessons
learned from research in Artificial Intelligence (AI) domain, entire replacement of human agents by machine agents, as decision makers, is not feasible particularly in areas as complex and knowledge intensive as engineering design. Rather, an effective agent-based system in engineering design should essentially serve as a decision support system for human designer and enable true collaboration between human and computer during product realization process. The major challenges associated with the development of a multi-agent system for distributed fixture design include semantic unification of the participating agents, integration of various design and analysis tools used in the domain of fixture design, and selection of appropriate architecture both at the agent level and at the system level. This chapter mainly focuses on system architecture, agent communication, and task allocation.

The remainder of this chapter is organized as follows. Next section reviews the major concepts and terms in agent technology followed by an overview of the existing works in agent-based engineering design. After a brief introduction to fixture design process, two aspects of designing an agent-based system for fixture development, namely, agent architecture and system architecture are discussed. The chapter ends with a conclusion and a brief discussion on future research directions in this area.

MULTI-AGENT SYSTEMS: AN OVERVIEW

This section provides a brief overview of major terms and concepts in agent technology. In the previous section, agent and multi-agent systems were defined very broadly. Since there are different viewpoints for defining agents in computer science community, it is beneficial to review several definitions by leading researchers. Brenner (Brenner, 1998) defines an agent as a software program that acts on behalf of its user and possesses some level of intelligence that permits it to perform some of its tasks autonomously and to interact with its environment in a useful manner so that it can achieve its pre-specified goals. Russell and Norvig (2002) define agent, from a more generic perspective, as anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. In Franklin and Grasser’s (1996) view, agent is an entity that senses its environment and acts on it over time in pursuit of its own agenda and so as to affect what it senses in the future. Some of the key concepts evident in these definitions are agent’s ability to interact with its environment, goal-oriented nature of agents, and possession of some level of intelligence. In this chapter, agents are always regarded as computational information processing entities that are consistent with the aforementioned definitions. A multi-agent system is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capabilities or knowledge of each agent (Brenner et al, 1998). In order to interact successfully, agents in a multi agent system require the ability to coordinate, cooperate, and negotiate with each other. Some of the core attributes of agents are autonomy (acting independently without human intervention), reactivity (perceiving the environment and responding the changes in a timely manner), pro-activeness (acting in anticipation of future goals) and social ability (ability to collaborate and interact with other users). Other properties that agents can posses include mobility, adaptability, veracity and, rationality. There are two types of problems that should be dealt with in design of a multi-agent system: one is agent design and the other one is society design. When designing the internal architecture of an agent, the goal is to equip the agent with the required components for successful fulfillment of the delegated tasks. Internal architecture of an individual agent describes the internal modules of the agent and the way they interact with each other. Some of
the internal modules commonly used in configuring an agent include the user interface, domain knowledge, and social knowledge, problem solving models, and coordination and scheduling modules (Shen et al, 2001). In designing a society of the agents, the intention is to enable agents to interact with one another especially when the other agents cannot be assumed to share the same interests and goals.

AGENT-BASED SYSTEMS FOR ENGINEERING DESIGN

In engineering design, agents are primarily used for enabling systematic cooperation among dispersed design teams, intermediating between legacy systems and connecting them semantically, and allowing better modularization and management of design process (Shen et al, 2001). Most of the existing agent-based systems are primarily geared toward the common goal of enabling intelligent concurrent engineering through facilitating distributed decision-making, streamlining interaction and coordination among designers and improving information exchange during the design process. Agent based systems in engineering design vary based on several criteria such as domain of application, agent structure, system architecture and communication language and protocols. Some systems are designed around a particular product (Cutkosky et al., 1993; Toye, 1994; Park, 1994) while some other systems adapt a more generic approach and target agent-based concurrent engineering from a more holistic perspective (Shen and Brathes, 1997; Parunak et al., 1999; Brown et al., 1995).

System architecture (i.e., the structure of control and collaboration relationship between individual agents in an agent society) is another criterion that differentiates various agent-based systems. System architecture varies in a continuum form fully centralized hierarchical architecture to fully decentralized autonomous architecture. Due to relative ease of development and maintenance, hierarchical architecture is widely adopted in industrial applications and several systems such as ADEPT (Norman et al., 1997) have been developed accordingly. However, several problems, such as lack of flexibility and reconfigurability, associated with the centralized nature of hierarchical systems have urged researchers to depart from centralized control and adopt federated architecture in several areas. Federated architecture combines centralized decision making and control with local autonomy. In agent-based concurrent design, several variations of the federated architecture, such as the facilitator approach (McGuire, 1993), the broker approach (Park, 1993), and the matchmaker approach (Decker, 1997), are implemented by different researchers.

Agent architecture is another distinctive dimension of multi-agent systems in engineering. Depending on the number and the complexity of the modules within agents, they range from very simple, single function agent to very complex, intelligent agents. In agent-based systems for concurrent engineering, those agent who are directly involved in engineering problem solving have the most complex internal architecture. For example, in SiFAs (Brown, 1995) an agent performs a single function (such as material selection from a strength point of view) whereas, in DIDE (Shen, 1995), agents are cognitive entities with deductive storage capabilities that participate in various design tasks.

Agent communication, as another key feature in any agent-based system, is addressed differently by exiting agent-based systems for engineering design. In some systems agent communication is
conducted simply through e-mails while, more sophisticated systems rely on Agent Communication Languages (ACL) based on Speech Act Theory such as KQML and FIPA as messaging protocols and use formal ontologies based on Knowledge Interchange Format, KIF, (Genesereth, 1992) and Description Logic, DL, (Baader, 2003) as the content language (Cutkosky et al., 1993; Toye, 1994). Choice of communication protocol and knowledge exchange language depends on the complexities of agent interaction model as well as the complexity of the generated and exchanged knowledge. Literature survey revealed that use of formal ontologies for knowledge representation is rapidly becoming an indispensable necessity in agent-based systems for engineering design.

Agent based concurrent engineering systems are differentiated based on some other factors such as task decomposition and allocation as well as agent negotiation and conflict resolution. Interested reader is referred to Shen et al. (2001) for an in-depth discussion on the way the aforementioned issues are handled by different systems. Existing works in agent-based engineering design provide a reasonable demonstration of how multi-agent systems can support concurrent engineering through task decomposition, distributed reasoning, collaborative decision making and integration of engineering tools, to name a few. However, we believe that these early explorations of agent technology in engineering design, despite their significant contribution as the seminal works in the field of agent-based design, have not sufficiently materialized the real gains that can be obtained from agent-based approach. In particular, the existing works in agent-based engineering design are either ad hoc to specific application domains or, if generic, too conceptual to be applicable to real-life engineering design problems. That’s why, in practice, few industrial systems implement agent-based approach for engineering design. As mentioned by Lander (1997) about a decade ago, “the challenge [in MAS research] is to move from an appealing framework to an effective application”. To this end, it is necessary to explore the challenges of agent-based approach for design of various engineering products as the system requirements in MAS vary based on the complexities of products and their corresponding engineering design process.

This chapter focuses on application of agent technology in engineering design of fixtures. Previous works in agent-based fixture design have partially addressed some of the challenges in areas such as agent messaging (Xinhua et al., 2007) and human knowledge capture (Subramanian et al., 2001). The presented framework particularly addresses system architecture, agent communication, and task decomposition. Next section provides an overview of fixture design process.

FIXTURE DESIGN PROCESS

A fixture is a device that locates and holds workpiece in position during manufacturing processes (Rong et al., 2005). Fixtures can be classified into two broad categories, dedicated fixtures and modular fixtures (Hoffman, 1991). Dedicated fixtures are commonly used in mass production and they are customized based on specific fixturing requirements of a particular product. Flexible (or modular) fixtures, on the other hand, are designed and fabricated based on standard fixturing elements such as standard clamps, locators and supports that can be assembled into a variety of configurations for accommodating different workpieces. Due to the inherent flexibility and adjustability of the modular fixturing systems, they are extensively used in low-volume production where manufacturing setups are subject to frequent change and reconfiguration (Rong
et al., 2005). The agent-based system proposed in this work is based on the requirements of modular fixturing systems.

Design of fixtures, both dedicated and modular, entails three major phases, namely, problem definition, fixture synthesis and fixture analysis (Bi and Zhang, 2001). The problem definition phase includes understanding the fixturing requirements and defining design variables (such as the variables which define the layout and configuration of the fixture), design constraints (such as closure, accessibility and ease of loading and unloading), and design criteria (such as cost of manufacturing and level of labor required) (Pehlivan and Summers, 2006). To define a fixturing problem, the problem formulating agent particularly needs information on the machining forces, the tool path, available surfaces for holding, material properties of the stock, the tolerance and surface finish requirements, and available fixturing elements. Fixture synthesis, as the next step in design, determines the values of design variables for a fixture configuration that can satisfy the design constraints. Regardless of the method used for fixture synthesis, the information generated in this phase includes the number, types, locations, and orientation of the fixturing elements. Both problem formulation phase and fixture synthesis phase require information of different types of fixturing elements. This information is usually available in a fixturing inventory, either electronic or paper-based, which provides detailed information on available locators, clamps and supports. Fixture analysis, as the final phase of fixture design, verifies the validity of the synthesized fixture with respect to design criteria (Yah and Liou, 2000). Typical approaches for fixture analysis are geometric analysis, kinematic analysis, force analysis, and deformation analysis. The input for fixture analysis generally includes the fixture configuration to be evaluated along with applied forces (i.e., inertial, gravitational, machining, and clamping forces) and the material properties of the stock.

Due to the complexity of fixture design process and its associated costs, which account for %10-20 of overall costs of manufacturing system (Bi and Zhang, 2001), automation of fixture design process through application of CAFD systems is becoming a common practice in industry. However, automation, by itself, is no longer the ultimate solution for the challenges of modern product development era. Next-generation CAFD systems need to be able to collaborate with each other in a heterogeneous and distributed environment. At the same time, they need some level of intelligence that enables them to serve as decision-support for human designer. Other requirements for next generation CAFD systems include modularity, reconfigurability, and extensibility.
Agent-based approach for fixture design can fulfill the aforementioned requirements by providing a shared computational framework for distributed problem solving in fixture design. Next section takes a closer look at agent-based fixture design by reviewing a fixture synthesis technique at an agent level. This section is followed by an in-depth demonstration of agent-based framework at a system level.

AGENT ARCHITECTURE: EVOLUTIONARY ALGORITHMS FOR FIXTURE DESIGN

As mentioned earlier, one of the major modules of agent’s internal architecture is the problem solving module. This module contains the algorithms and heuristics required for fulfilling agent’s intended goals. In this section a representative agent-based system, called multi-agent fixture design system or MAFDS (Subramaniam et al., 2001), is discussed in order to help readers gain a better understanding of the problem solving strategies adapted in agent-based fixture design systems.

MAFDS has two main modules: fixture designer and fixture evaluator. A fixture design problem in MAFDS is defined through workpiece information and process information (Table 1 – columns a and b). Last column in Table 1 lists the components of a fixture design solution that collectively form the solution space for a given fixturing problem. Different combinations of these components specify a particular fixturing solution.

![General Architecture of MAFDS](image)

*Figure 2: General architecture of MAFDS*

*Fixturing designer* module uses genetic algorithm (GA) to search through the solution space. GA is reported as an efficient search method for arriving at optimal and near-optimal solutions in a large and complex solution space such as fixture design solution space (Arena, et al., 1993). Each fixturing solution returned by the fixture designer is represented as a chromosome and each component of fixturing solution (listed in Table 1) is regarded as a gene. The fixture designer selects chromosomes to be reproduced based on their corresponding fitness value. Using genetic
operators such as mutation and crossover, new fixture designs (chromosomes) with better fitness value are evolved and the cycle continues until the desired fitness value is obtained. One of the advantages of GA over other optimization methods, such as linear programming, is that it can yield a set of sub-optimal solutions as well. As a result, human designers can explore different design alternatives and select the most appropriate one as the skeletal design and arrive at the final solution through further design iterations.

**Fixture evaluator** measures the fitness value of individual chromosomes using neural network heuristics based on three major criteria, namely, ease of loading and unloading, cost and rate of production. The obtained fitness values for each fixture design are fed back to the fixture designer to be used in selection and reproduction process. The training data for the neural network is created through a performance matrix that relates the input and output of the fixture evaluator. For example, this matrix specifies how different variants of a base locator (i.e., flat with slots, flat, and 3 points) affect the final performance measures. The performance matrix, in fact, can be regarded as the domain knowledge of the design agent. Domain knowledge, as another integral component of an intelligent agent, encapsulates the description of working projects (problems to be solved), partial states of the current project, rules and hypotheses developed and intermediate results etc. The domain knowledge developed in this work is confined by the parameters that define the solution space. However, to develop a more comprehensive knowledge base, it is necessary to further expand the design space and extract more credible rules from real industrial cases.

| **Table 1: Workpiece, process and fixture information as used in MAFDS** |
|------------------|------------------|------------------|
| **Workpiece information** (a) | **Process information** (b) | **Fixture information** (c) |
| Weight | Batch size | Base locators |
| Size | Spindle speed | Side locators |
| Pre-machined features | Operation type | End locators |
| | Base support surface | Clamping directions |
| | Primary locating surface | Type of clamp |
| | Secondary locating surface | Clamp actuation |
| | | Fixture body |

To validate the proposed system, it was applied to the design of a fixture used in the process of drilling hole # 1 in the “vibrator arm” shown in Figure 3. Workpiece and process information for the vibrator arm is shown in Table 2. As can be seen in Table 3, the designed obtained by the agent-based system is sufficiently close to that of the human designer as they have five components in common. The major shortcoming of the proposed system is that the design space is pretty much limited by the design inputs. To improve the quality of designed fixtures, it is necessary to provide more detailed information about the workpiece, process and even manufacturing equipments that are used during the machining operation. In other words, a larger number of design variables need to be included in the definition of solution space. Nevertheless, the
The proposed methodology provides a realistic demonstration of how agents can be programmed for the purpose of designing fixtures.

**Figure 3: A vibrator arm**

**Table 2: Workpiece and process information for the vibrator arm.**

<table>
<thead>
<tr>
<th>Input type</th>
<th>Input variant</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>Medium</td>
<td>1</td>
</tr>
<tr>
<td>Spindle speed</td>
<td>Fast</td>
<td>1</td>
</tr>
<tr>
<td>Operation type</td>
<td>Drilling</td>
<td>1</td>
</tr>
<tr>
<td>Part weight</td>
<td>Light</td>
<td>0</td>
</tr>
<tr>
<td>Part size</td>
<td>Medium</td>
<td>1</td>
</tr>
<tr>
<td>Pre-machined feature</td>
<td>Step</td>
<td>0</td>
</tr>
<tr>
<td>Base support surface</td>
<td>Plain</td>
<td>1</td>
</tr>
<tr>
<td>Primary locating surface</td>
<td>Plain</td>
<td>1</td>
</tr>
<tr>
<td>Secondary locating surface</td>
<td>Plain</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 3: Comparison of fixturing solutions**

<table>
<thead>
<tr>
<th>Fixturing components</th>
<th>Fixturing Solution by Human designer</th>
<th>Fixturing Solution by MAFDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base locators</td>
<td>3 points</td>
<td>Flat with slot</td>
</tr>
<tr>
<td>Side locators</td>
<td>Flat</td>
<td>Flat</td>
</tr>
<tr>
<td>End locators</td>
<td>Flat</td>
<td>Flat</td>
</tr>
<tr>
<td>Clamping direction</td>
<td>Two sides</td>
<td>2 sides</td>
</tr>
<tr>
<td>Type of clamp</td>
<td>Toggle</td>
<td>Toggle</td>
</tr>
<tr>
<td>Clamping actuation</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>Fixture body</td>
<td>Built-up</td>
<td>Welded</td>
</tr>
</tbody>
</table>
SYSTEM ARCHITECTURE: AN AGENT-BASED FRAMEWORK FOR FIXTURE DESIGN

As discussed earlier, the major sub-tasks in the development of a multi-agent system include developing agent architecture, selecting an approach for agent organization, selecting or developing languages and protocols for inter-agent communication, selecting or developing protocols and mechanisms for cooperation, coordination and negotiation and developing the required mechanism for task decomposition and allocation. In the previous section, a specific agent architecture for fixture design based on genetic algorithm and neural network was discussed. This section focuses on system architecture through introducing a multi-agent framework for fixture design called AFFIXED. Also, agent communication and task decomposition and allocation are investigated in the context of the proposed framework.

System Architecture:

System architecture of a multi-agent system essentially describes the agent types that participate in the system as well as the pattern of relationships among the agents. As mentioned earlier, system architecture in agent-based systems can be classified into three major categories: hierarchical architecture, federated architecture and autonomous architecture. Federated architecture, due to enabling partial decentralization, is the most appropriate architecture for developing complex and dynamic multi-agent systems in which numerous distributed agents with different capabilities are involved in delivering the objectives of the system (Shen et al., 2001). Accordingly, a federated architecture is proposed for AFFIXED since fixture design is a complex process which requires a wide range of capabilities provided by various resources.

Figure 4 shows the proposed federate architecture for AFFIXED. As can be seen in this figure, there exist 8 types of agents in the proposed architecture: requirement directory (RD), problem definition (PD), fixture synthesis (FS), fixture analysis (FA), project manager (PM), definition coordinator, (DC), synthesis coordinator (SC), and analysis coordinator (AC). PD, FS and FA are remote agents while RD, DC, SC, AC and PM are the local agents which collectively form an AFFIXED project platform. In the proposed system, each project platform can handle several fixturing projects. Also, to enable scalability, several instances of project platform can be instantiated in the multi-agent system. The aforementioned agents are described below:

Requirement Directory Agent (RD): Requirement Directory agent is in charge of collecting fixturing requirements (such as part geometry and material, manufacturing equipments, process plan, etc.) from the customer and maintaining a list of active requirements. RD agent provides a user interface for the human users. Through RD’s interface, human users enter the requirements of the fixturing problem and submit their request for design. Project Manager (PM) agent regularly visits the requirements list and initiates a fixturing project for each set of fixturing requirements.

Problem Definition Agent (PD): The main function of PD agent is to define the fixture design problem in terms of a set of design variables, design constraint, and design criteria as defined in section 0. There are various approaches for formulating a set of fixturing requirements into a fixturing problem. Therefore, several PD agents can participate in a fixture design project and define the problem from their own perspective. A cluster of PD agents is connected to the PM agent through a coordinator (facilitator).
Figure 4. The federated architecture for agent-based fixture design

Fixture Synthesis Agent (FS): FS agents are remote agents who are in charge of actual design of the fixture based on the formulated fixturing problem. Some of the subtasks of fixture synthesis process include selection of types of fixturing elements (i.e., locators, clamps, supports, etc.), specification of locating and supporting points, determination of clamping sequence and well as assembly planning of the fixture. Given the computational complexities of fixture synthesis process, several FS agents might concurrently participate in this process each addressing a subset of entire design process. For example while one FS agent may be expert in determining the clamping and locating points, another agent may be specialized in calculating clamping forces to minimize work piece deflection. Furthermore, several agents might perform the same design sub-task differently adopting different problem solving methods (such as GA and NN as discussed in the previous section), thus resulting in alternative fixture designs.

Fixture Analysis Agent (FA): The primary function of FA agents is to verify fixture design against the design constraints. Different types of analysis include kinematical analysis, force analysis, geometric analysis, and deformation analysis. The input used by fixture analysis agents typically includes the configuration of the fixture under evaluation, applied forces as well as the material properties of the work piece. FA agents might be interfaced with FEA systems as FEA is one of the primary methods used during fixture analysis phase.

Definition Coordinator Agent (DC): A DC agent serves as the facilitator for the problem formulation phase. In some scenarios, a DC agent has the authority to select the most suitable problem from a set of problems formulated by various remote PD agents. Otherwise, it passes the problem set to the project manager agent which will select the appropriate problem based on some pre-defined criteria. PD agents, to be considered in the fixture design project, must register themselves in advance with the DC agent.
Synthesis Coordinator Agent (SC): The primary tasks of SC agent include registering the FS agents based on their specialty and notifying them upon arrival of a new fixture design problem. Also, SC agent integrates different pieces of the design submitted by different FS agents into various complete fixturing solutions.

Analysis Coordinator Agent (AC): AC agent is in charge of coordinating fixture analysis process. Same as SC agent which serves as the directory of fixture synthesis agents, AC agent provides a listing of FA agents grouped based on the type of analysis task. AC provides FA agents with the required information and collects the analysis results from various FA agents.

Project Manager Agent (PM): PM agent functions as the manager of the entire fixture design project. Without coordination, the proposed agent based system will quickly degenerate into a chaotic collection of individual agents. PM receives the fixturing requirements from the requirement directory agent and initiates a fixturing project. PM interact with the facilitator agents in problem definition, fixture synthesis and fixture analysis. An important role of PM agent is to interact with human designer during the design process. As mentioned earlier, since the proposed system is not intended to automate the design process thoroughly, interaction with human designer is an integral feature of the agent-based system proposed in this work. Figure 5 shows the interaction model of agents in AFFIXED.

Agent Communication:
Communication is a key enabler of coordination and collaboration in any multi-agent system. Communication standards and protocols are the essential components of agent communication system. There are two types of communication standard that should be supported by agent-based systems: one at messaging level and the other one at content level.

In the messaging level, standard Agent Communication Languages (ACLs) provide the required protocols and standards for exchanging message between agents. Most of the existing agent-based systems in engineering design use KQML as the ACL. Recently, FIPA\(^1\) has become a more popular ACL due to its enhanced information exchange capabilities at the semantic level. In this work FIPA is used as the agent communication language. In FIPA-ACL, each message is structured through a set of pre-defined elements such as performative, sender, receiver, reply-to, content, language, ontology and protocol. Performative, as the only mandatory element in any ACL message, denotes the type of the communicative act of the message. For example, the Project Manager (PM) agent, when requesting fixture requirements from the Requirement Directory (RD) agent, uses REQUEST as the performative. FIPA-ACL does not mandate any particular language for content specification but it imposes that the content language must be able to express propositions (states that some sentence in a language is true or false), objects and actions.

---

\(^1\) [http://www.fipa.org/](http://www.fipa.org/)
In the content level, an ontological approach is adopted in AFFIXED. In a distributed and heterogeneous agent-based system, common syntax and semantics for content representation is an absolute necessity. Particularly, since engineering design is a knowledge intensive process, rich representation of engineering knowledge through a formal ontology is critical to the performance of the multi-agent system for fixture design. To enable intelligent decision making by the agent society, the ontology language needs to have enough expressive power for encoding a wide spectrum of knowledge ranging from design specifications and constraints to design rules and axioms. In this research, an ontology, called FIXON, is developed based on Description Logics (DL) for content representation. DL provides a formal syntax and semantics for describing knowledge and developing information models within a domain in terms of concepts and properties that specific individuals must satisfy. DL offers several advantages over other ontology representation languages. First, DL is equipped with a formal, logic-based semantics (Baader, 2004). Therefore it allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. Typical reasoning tasks in DL include subsumption, equivalence, and satisfiability checking. Second, DL is supported by the Semantic Web. Therefore, engineering information can be shared through web-based technologies and tools such as browsers and the eXtensible Markup Language (XML). Accordingly, researchers have recently shown a growing interest in the development of DL-
based engineering ontologies (Kopena and Regli 2003; Lee and Suh 2005; Ameri and Dutta 2006; Udoyen 2006; Mocko, Rosen et al. 2007).

Figure 6(a) shows some of the core concepts and relationships in FIXON. DL enables formal definition of concepts through logical constraints represented as necessary and sufficient conditions.

**Figure 6. (a) Some of the core concepts and relationships in FIXON (b) A partial view of fixture component taxonomy in FIXON**

For example, clamp, can be defined as following:

Clamp ≡ fixtureComponent ∩ (∃ constrains . Workpiece) ∩ (∃ appliesForce . Force) (= 1 hasDirection.Direction) ∩ (∃ hasPosition . Position)

This definition states that Clamp is a fixture component with a fixed position that constrains the movement of workpiece through applying force in exactly one known direction. In this way, machine agents can understand the semantics of the term clamp and there is no need to hardcode clamp as a key word in the agent’s routines. There exist many other formal definitions similar to clamp’s definition in FIXON. Also, there exist several taxonomies in FIXON including fixture component taxonomy, material taxonomy, and process taxonomy. A partial view of fixture component taxonomy is shown in Figure 6(b). Using FIXON’s vocabulary, agents can generate and manipulate various types of FIXON descriptions including fixturing requirements, fixture problems, fixture solutions and fixture analysis results. Figure 7 (a) and (b), shows partial
description of a fixturing problem and a fixturing solution in FIXON respectively. Web Ontology Language (OWL\(^1\)) is selected as the ontology language of FIXON.

\[\text{Figure 7.} \text{ (a) Partial description of a fixturing problem through describing the machining operation (b) Partial description of a fixturing solution through describing one of the clamps that is designed into the solution.}\]

**Task Allocation:**
In general, there are two categories of task allocation methods in agent-based systems: centralized and distributed. In centralized allocation, a supervisor or facilitator agent is in charge of decomposing the tasks and allocating them to appropriate agents whereas, in the distributed allocation, individual agents independently search among their peers and allocate their corresponding tasks to a suitable agent (Shen et al., 2001). Allocation through bidding, for instance, is one of the common distributed allocating methods practiced in agent-based systems. In AFFIXED, a centralized allocation approach is adopted for task allocation as the coordinator agents are in charge of assigning fixturing tasks to remote agents. Bottleneck creation, a common problem in centralized system, is not a concern here because, as mentioned before, scalability in AFFIXED is achievable through instantiation of multiple project platforms. The granularity of tasks is at the project level. In other words, allocation of the major tasks of a fixturing project (i.e., problem definition, fixture synthesis, and fixture analysis) is main concern. Further breakdown of these tasks, although absolutely feasible and meaningful, is deemed outside the scope of AFFIXED project in its current status. In AFFIXED, two methods are developed for allocating tasks to the remote agents: 1) cluster-based allocation and 2) first-k allocation.

**Cluster-based allocation:**
Simply stated, through cluster-based allocation, a single task is assigned to all members of an agent cluster. In agent clustering, AFFIXED relies on the classification capabilities of DL-based

\(^1\)http://www.w3.org/2004/OWL/
reasoners. More specifically, since capabilities of AFFIXED agents are described using FIXON ontology, a DL reasoner can take a flat population of agents as the input and generate an inferred taxonomy of agents based on their formal description. Therefore, a coordinator agent, for instance, can assign a fixturing problem to a sub-category of a FS agent taxonomy based on different criteria such as the machining process type, fixture complexity or fixture type (dedicated or modular). For example, Figure 8 shows an agent cluster simply formed based on the type of service the agents provide. In this example, the service of interest is fixture synthesis. Therefore, the synthesis coordinator agent allocates the task to all three agents who provide this service irrespective of their other characteristics (such as fixture type expertise).

![Agent Cluster Diagram](image)

*Figure 8. An agent cluster composed of three agents (agent1, agent2, and agent3) along with their formal definition.*

**First-k allocation:**

In the first-k allocation method, an ordered list \((L_t)\) of active agents is generated at time \(t\) and the task is allocated to the first \(k\) agents in the list. Agents are ordered based on their overall score. In AFFIXED, there are two major factors which contribute in agent scoring: one is past performance and the other one is similarity with the current fixturing project. Accordingly:

\[
Score_i = w_p P_i + w_s S_i
\]

Where, \(P_i\) is the performance score of the \(i\)th active agent based on its previous record and \(S_i\) is the similarity score of the agent. \(w_p\) and \(w_s\) are the weighting factors which denote the relative importance of performance and similarity in the overall rating. The similarity algorithm used for calculating similarity score of the agents is an adaptation of the feature-based method which can be used for measuring the semantic proximity of concepts defined within an ontology (Tversky, 1977). This method uses formal definition of the concepts as the basis of comparison. According to the feature-based method, semantic similarity depends on both common and uncommon features used in the definition of the concepts. Therefore, similarity of concepts A and B can be calculated through the following equation:
\[ \text{Sim}(A, B) = \frac{n_{A \cap B}}{n_{A \cap B} + \mu n_{A-B} + \nu n_{B-A}} \]  

(2)

Where:

\[ n_{A \cap B} = \text{Number of features common to both A and B.} \]
\[ n_{A-B} = \text{Number of features that belong to A, but not B.} \]
\[ n_{B-A} = \text{Number of features that belong to B, but not A.} \]

In equation 2, the first term of comparison (i.e., A) is referred to as the **target** while the second term (i.e., B) is the **base**. \( \mu \) and \( \nu \) are the weighting factors which determine relative importance of distinctive features. If \( \mu = \nu \) then the matching in known to be symmetric which means features in both A and B are of equal importance. In an asymmetric matching, more emphasis is put on features of one concept (either A or B). Since agent’s capabilities in AFFIXED are described through a set of logical constraints (features), the feature-based method can be used for calculating the semantic similarities of agents. In equation 2, the target agent is the desirable agent for a given task (e.g., fixture synthesis) in a project and the base agent, is a remote agent that will be compared against the target agent. The upper part of Figure 9 shows the FIXON description of a desirable fixture synthesis agent (TargetAgent) and the lower part described the **BaseAgent**. Based on the description of the TargetAgent, the desirable synthesis agent is expert in designing fixtures for automotive industry and provides some fixture layout services. The fixtures designed by the desirable synthesis agent have at least have two elements (clamps and locators), have modular configuration (not dedicated), are designed for machining processes with tight tolerances, and support prismatic workpieces made of aluminum. If an agent with exact same features can be found among the remote agents, the synthesis task will be assigned to this agent. However, since exact matches seldom happen, partial mates are considered as well in task allocation. Based on equation 2, the semantic similarity between TargetAgent (A) and BaseAgent (B) is calculated as follows:

\[ \text{Sim}(A, B) = \frac{n_{A \cap B}}{n_{A \cap B} + \mu n_{A-B} + \nu n_{B-A}} = \frac{6}{6 + .8 \times 3 + 0.2 \times 7} = 0.61 \]

In this calculation, it is assumed that importance of the features that exist in A but not B is more than the importance of features that exist in B but not A, hence \( \mu = .08 \) and \( \nu = .02 \). In finding common features, subsumption relationship is taken into account. For instance, since the Milling process is a sub-class of the Machining process, the **hasProcess** features of A and B are regarded as identical. Logically, if an agent can is capable of designing machining fixtures, it should be able to support the design of milling fixtures.
Figure 9. DL representation of TargetAgent and BaseAgent

Through one-on-one comparison of the TargetAgent with all registered remote agents, a ranked list of remote agents based on their semantic similarity with the desirable agent can be generated and used for task allocation.

IMPLEMENTATION

The implementation of AFFIXED in its current status mainly includes development of agent communication and interaction modules together with the required user interfaces. FIXON ontology is implemented in Protégé\(^1\) and AFFIXED is implemented in JADE\(^2\) agent platform. JADE is a middleware that facilitates development of multi-agent systems. As a middleware, JADE provides higher-level libraries of Java classes that enable more effective development of agent-based systems by hiding the complexity and the diversity of lower-level hardware and software systems to the application developer. Also, JADE provides a run-time environment in which agents can live and interact with each other. Furthermore, JADE framework is equipped with a suit of graphical tools which allow administrating and monitoring the activities of running agents. JADE is compliant with FIPA (Foundation for Intelligent Physical Agents) specifications, a suite of standards for agent communication. Therefore, it enables interoperability among agents who participate in an agent society. Figure 10 shows a JADE agent container loaded with JADE platform agents and AFFIXED platform agents. In JADE, every agent is contained in a container. A JADE platform is composed of one or more active JADE containers that can be distributed over the network. An AFFIXED project platform is an

---

\(^1\) [http://protege.stanford.edu/](http://protege.stanford.edu/)

instance of JADE platform and consists of five agents as shown in Figure 4. An AFFIXED platform can be hosted on a single machine or on several distributed machines.

![Figure 10: JADE agent platform loaded with AFFIXED platform agents](image)

As mentioned earlier, AFFIXED is designed for supporting human designers during the fixture design process. Therefore, most of AFFIXED agents can interact with human users through one or more user interfaces. Figure 11 shows the user interface of the Project Manager agent for creating new fixture design project. As can be seen in this figure, fixture designer mainly specifies the termination rule for the project (by defining a limit for either the duration of the project or the number of solution that have to be generated by AFFIXED) and select a particular requirement set for the project. Once the project is created, PM agent takes the next step by informing the Definition Coordinator (DC) agent of the initiation of a new project. DC agent then submits the fixturing requirements, represented in FIXON, to the registered problem formulation agents which, collectively, generate multiple fixturing problems based on different criteria. PM (or DC) agent then selects an appropriate fixturing problem and submits it to the SC agent which allocates the synthesis task to the qualified fixture synthesis agents (based on the allocation strategies describe earlier). Figure 12 shows the user interface of the synthesis coordinator agent which provides listings of registered FS agents, past projects, and current active projects. As can be seen in this figure, remote FS agents are accessible through their JADE container’s address.
Figure 11: PM user interface for creating a new fixture design project

Figure 12: The user interface of the SC agent

Fixture analysis (FA) agents are the last set of agents which get involved in the project through the analysis coordinator (AC) agent. Figure 13 shows the message sent back by a FA agent to the AC agent about the results of a deflection analysis on the given design. As can be seen in the detailed view of the FIXON message, two forces (one clamping force and one machining force) cause deflections beyond the allowable value.
CONCLUSION AND FUTURE DIRECTIONS

This chapter provided a high-level discussion on how multi-agent systems can support fixture design process in distributed environments. Given the multiplicity of possible solutions for a fixture design problem as well as distribution of knowledge in the open, decentralized and, dynamic design environment envisioned for fixture development, fixture design problem is within the natural domain of multi-agent systems. Design of agent-based system poses two types of problems one at an agent level (micro) and one at a system level (macro). Although the agent level problem (i.e., how to enable an individual agent solve a given problem locally) is not a trivial one, it is less challenging as compared to the system level problem mainly due to the size, complexity, and heterogeneity issues. In this chapter, an example of agent architecture for fixture design based on evolutionary algorithms was selected from the literature and reviewed just to familiarize the readers with the nature of problem solving in the fixture design process. Through this example, it was shown that fixture design, like several other design processes, is mainly performed by valuation of a fixed set of design variables and evaluation of the resulted design based on predefined measures of goodness. A stand-alone CAFD system can be regarded as a computational agent that is tasked with fixture synthesis. If the entire fixture design problem can be properly solved by a single CAFD system, a multi-agent approach is not justifiable. The real benefits of agent-based system in fixture design reveal themselves when the fixturing problem is too complex to be addressed by a single CAFD system.

The main focus of this chapter was on the system level design. Literature survey showed that existing works in agent-based engineering design are either ad hoc to specific application domains or, if generic, too conceptual to be applicable to real-life engineering design problems. Therefore, to fill the identified gap, AFFIXED is proposed as an agent based system particularly developed for fixture design. Development of AFFIXED is also motivated by the need for a computational infrastructure that enables integration of disparate CAFD systems in distributed environments. AFFIXED is designed at the outset for providing decision support for human designers during the fixture design process. Therefore, availability of suitable user interfaces that
facilitate effective interaction between human users and machine agent is pivotal to the overall performance of the developed system. Although AFFIXED can support a complete fixture design project, human designers can use AFFIXED for finding solutions for partial fixturing problems. One outstanding feature of AFFIXED is incorporation of a formal logic-based ontology for knowledge representation which enables communication of distributed agents at a semantic level. In its current state of implementation, AFFIXED treats remote agents as black boxes that have encapsulated fixture design tools and expertise. Future work in development of AFFIXED includes creation of different types of remote agents (i.e., problem formulation, fixture synthesis and fixture analysis) with different expertise and capabilities in order to build a more realistic testbed for examining various system architectures and interaction models.

Future research direction in agent-based fixture design is closely coupled with the major trends in evolution of industrial information systems. For decades, integration has been regarded as the holy grail of industrial enterprises and several technologies have been developed to enable several industrial systems to operate as an integrated entity. Despite significant achievements in this area, integration remains to be a major research issue in development of industrial systems. Since fixture design systems are located at the interface of design and manufacturing, they need to be carefully supported by the required IT components that enable integration. Among the enablers of integration, information exchange standards have a critical role in seamless connection of fixture design systems with the surrounding systems including CAD and computer-aided process planning (CAPP). Given the fact that information interoperability is still an open problem in industrial information systems, it is envisioned that development of communication standards continues to be an active field of research in this domain. In particular, development of ontological information models, due to their semantic capabilities, will be an area of interest.

Another visible trend in evolution of industrial systems is moving toward a higher degree of intelligence, autonomy, and adaptability in order to depart from centralized and hierarchical control systems. Agent-based systems, Holonic Manufacturing Systems (HMS), and Biological Manufacturing Systems (BMS) are among the decentralized and hierarchical solutions suggested by researchers for the next-generation industrial systems. Also, some other researchers have fostered further application of AI in industrial systems with goal of improving their adaptability to changes in the environment and expanding their scope of autonomous decision-making. Cognitive Technical Systems (Zaeh et al., 2007), for instance, is a solution that suggests application of embedded sensor and actuators in physical systems in order to enable them to perceive, learn, reason, and plan without human supervision. A Cognitive Factory, is a special instance of the cognitive technical system composed of different manufacturing resources including, work cells, machines, tools, robots, storage equipments each represented by an intelligent agent and equipped with sensing and actuating capabilities. More precisely, in a cognitive factory, planning intelligence is embodied in the physical components of the factory. Fixtures, in the context of the cognitive factory, need to be able to acquire product and process information in a real-time fashion, communicate their current status (in terms of configuration and work plan) with other production resources, and react to changes in the environment appropriately (through reconfiguration for instance). To this end, there is a need for development of fixturing knowledge models, autonomous planning capabilities, perception and
control mechanisms, and a cognitive perception-action loop that enables real-time fixture planning, design, and reconfiguration.

References


