CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

The Federation of Collaborative Design Agents

By

T. Khedro and P. Teicholz

CIFE Technical Report #113
October, 1997

STANFORD UNIVERSITY

If you would like to contact the authors please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

SUMMARY

CIFE TECHNICAL REPORT # 113

Title: The Federation of Collaborative Design Agents

Authors: Taha Khedro and Paul Teicholz

Publication Date: October, 1997

Funding Sources:

- CIFE Seed Funding
- US Army Construction Engineering Research Laboratory (CERL)

1. Abstract:

The Federation of Collaborative Design Agents (FCDA) is an agent-based collaborative design framework that was developed as a part of a CIFE research project which started in early 1991. The initial objective of this project was to explore the use of agent-based architectures for the integration of distributed software applications in support of collaborative facility engineering. The initial development of the FCDA framework was carried out in cooperation with Michael Genesereth of Computer Science. His work on agent/facilitator architecture and its application to automated factory called Designworld provided the initial computing foundation for the FCDA framework [Genesereth 91] [Genesereth 94].

The research effort on the FCDA project has progressed in three phases. The first phase built on the concepts of agent architectures to develop the FCDA framework focusing on building design and construction planning. This phase resulted in the definition and development of the FCDA framework as well as a prototype environment for collaborative distributed building engineering called CIFE WORLD [Khedro 94a]. The second phase of the project focused on collaboration issues among designers working in a geographically-dispersed environment. For this phase, a distributed collaborative CAD-based environment was developed and called AgentCAD [Khedro et al. 95]. This environment has served as a testbed in a project-oriented Stanford course on Integrated AEC [Fruchter 96a]. The third and final phase of the FCDA research effort involved four major universities in a funded project by the Construction Engineering Research Laboratory of US Army Corps of Engineers (USACERL) called Agent Collaboration Language Project (ACL Project). The objective of the ACL Project was to develop an Internet-based testbed for facility design and energy analysis. The testbed integrated independently-developed systems by Carnegie Mellon University, Massachusetts Institute of Technology, University of Illinois at Urbana-Champaign, and USACERL [Khedro et al. 95]. The cooperative research and development of the ACL Project testbed have examined and refined some aspects of the agent-based architecture supported by the FCDA framework.

2. Subject:

This research investigates a new approach for interactive design of facilities using a software architecture called The Federation of Collaborative Design Agents (FCDA). This is an agent-based architecture that uses one or more facilitators (a special kind of agent) to interpret the messages

being sent among the design agents and identify which other agents need the information in a given message. For each agent that needs this information, an appropriate message is sent to that agent (if it is on line and registered with the facilitator). Using this approach, it is possible for a group of designers at different locations to work together in a collaborative approach without the constraints of using the same software or database.

3. Objectives/Benefits:

The objective of the ACL Project was to develop an Internet-based testbed for facility design and energy analysis. This is a new approach for distributed, concurrent design that eliminates the need for each designer to use the same design program or to share a common database. All that is needed is that each design agent use an object-oriented approach (not vector-based) and that these objects be defined in a central repository (the facilitator) so that they can be recognized and interpreted by the facilitator. This approach is much more general that alternatives known to the authors and provides the basis for very significant enhancements in design productivity.

4. Methodology:

The research effort on the FCDA project has progressed in three phases. The first phase built on the basic concepts of agent architectures to develop the FCDA framework focusing on building design and construction planning. This phase resulted in the definition and development of the FCDA framework as well as a prototype environment for collaborative distributed building engineering called CIFE WORLD [Khedro 94a]. The second phase of the project focused on collaboration issues among designers working in a geographically-dispersed environment. For this phase, a distributed collaborative CAD-based environment was developed and called AgentCAD [Khedro et al. 95]. This environment has served as a testbed in a project-oriented Stanford course on Integrated AEC [Fruchter 96a]. The third and final phase of the FCDA research effort involved four major universities in a funded project by the Construction Engineering Research Laboratory of US Army Corps of Engineers (USACERL) called Agent Collaboration Language Project (ACL Project). The objective of the ACL Project was to develop an Internet-based testbed for facility design and energy analysis. The testbed integrated independently-developed systems by Carnegie Mellon University, Massachusetts Institute of Technology, University of Illinois at Urbana-Champaign, and USACERL [Khedro et al. 95]. This cooperative research and development of the ACL Project testbed allowed us to refine some aspects of the agent-based architecture supported by the FCDA framework. It also provided some "real world" feedback regarding the strengths and weaknesses of this approach.

5. Results:

This research project produced a prototype testbed that implemented the FCDA architecture using four agents (architectural design, structural design, energy analysis, facility requirements and visualization) to work together over an Internet (TCP/IP) connection. These agents communicate with a central Facilitator program which then interprets their ACL messages and sends the appropriate messages to each relevant agent on the network. Using this approach, a simple building (cabin) was designed. The project developed application program interfaces (APIs) for each agent that allowed them to communicate with the facilitator using the knowledge interface format (KIF) language. This approach was evaluated by the joint project team and the results are reported in this research report.

6. Research Status:

This research is completed. The prototype software is available to other CIFE researchers for further study and development. Desirable enhancements include the ability to negotiate changes within the system, the ability to capture the rational for design changes and additions so that these rational are available to the project team and the client to facilitate future changes. Another important area for research is control over design evolution, capturing versions of a design, etc. This issue is a necessary part of a distributed design environment, and has not been addressed in this research project.

The Federation of Collaborative Design Agents

by Taha Khedro[1] and Paul Teicholz[2]

0 Introduction

The Federation of Collaborative Design Agents (FCDA) is an agent-based collaborative design framework that was developed as a part of a CIFE research project which started in early 1991. The initial objective of this project was to explore the use of agent-based architectures for the integration of distributed software applications in support of collaborative facility engineering. The initial development of the FCDA framework was carried out in cooperation with Michael Genesereth of Computer Science. Michael Genesereth's work on agent/facilitator architecture and its application to automated factory called Designworld provided the initial computing foundation for the FCDA framework [Genesereth 91] [Genesereth 94].

The research effort on the FCDA project has progressed in three phases. The first phase built on the concepts of agent architectures to develop the FCDA framework focusing on building design and construction planning. This phase resulted in the definition and development of the FCDA framework as well as a prototype environment for collaborative distributed building engineering called CIFE WORLD [Khedro 94a]. The second phase of the project focused on collaboration issues among designers working in a geographically-dispersed environment. For this phase, a distributed collaborative CAD-based environment was developed and called AgentCAD [Khedro et al. 95]. This environment has served as a testbed in a project-oriented Stanford course on Integrated AEC [Fruchter 96a]. The third and final phase of the FCDA research effort involved four major universities in a funded project by the Construction Engineering Research Laboratory of US Army Corps of Engineers (USACERL) called Agent Collaboration Language Project (ACL Project). The objective of the ACL Project was to develop an Internet-based testbed for facility design and energy analysis. The testbed integrated independently-developed systems by Carnegie Mellon University, Massachusetts Institute of Technology, University of Illinois at Urbana-Champaign, and USACERL [Khedro et al. 95]. The cooperative research and development of the ACL Project testbed have examined and refined some aspects of the agent-based architecture supported by the FCDA framework.

This paper discusses various aspects of the FCDA framework including the concept of design and software agents, the message-oriented communication model, the modeling design of vocabularies for communications, and the federated agent/facilitator architecture. Then, we describe the three environment prototypes: CIFE WORLD, AgentCAD, and ACL Project testbed. Finally, the paper concludes with a brief summary and conclusions about this research.

---

[1] Research Associate, CIFE

[2] Professor (Research) CE, Director, CIFE

1 Background

Designers deliberately group together to engage in the goal-oriented activity of designing an agreed artifact. They collaborate to perform tasks requiring more than one person. Inherent in such collaborative work is a set of problems which require the communication and coordination among designers who use various types of engineering software applications (e.g., CAD systems, analysis programs, etc.) in a geographically-dispersed environment. Addressing these problems involves the integration of engineering software applications through which communication and coordination among designers are achieved. There have been three main challenges facing any software integration framework:

(1) inherent diverse semantic representations between captured design models,

(2) heterogeneous implementations and interfaces of engineering software, and,

(3) inadequate (limited) file-based support for exchange of design information.

Early integration approaches have proposed the use of database management systems (DBMS) as the primary integration scheme. These approaches have promoted query/update oriented communication models in which engineering applications interface directly with a database using a query-based language (e.g., SQL). Most database centric approaches presume that engineering applications can agree on a broad design product model or schema from which specialized design views can be derived according to applications' specific tasks. It is our view that such approaches do not provide adequate integration solutions from three perspectives. First, the supported communication model is limited to database updates and queries and does not respond to other needed forms of designers' interaction (e.g., direct and indirect communications). Second, database centric integration architectures require a close coupling of communication (i.e., data updates and queries) with the product model leading to an inflexible architectural solution. Third, database schemas tend to be mostly static which does not adequately support the dynamic nature of the engineering design process and its software. Typically, design product models used by engineering applications cannot be treated as static or frozen. They have to be modified and extended frequently, requiring existing version of the model to be migrated from one version to another and demanding accommodation of new design.

The FCDA framework is based on the concept of a Design Agent defined as a collection of Software Agents and a human designer. Software agents represent design and engineering applications that communicate via a networked infrastructure of coordination system programs called Facilitators. These facilitators use a message-based language called Agent Communication Language (ACL). Under a message-oriented communication model, facilitators provide a number of services aimed at routing and translating information based on different design models captured by software agents in a distributed environment. Unlike most database integration approaches, software agents communicate design information according to their different product models, and facilitators translate among them. The FCDA framework provides for a loosely-coupled federated system architecture that adapts existing engineering applications on different computing platforms as software agents through Agent Programmatic Interfaces (APIs). After discussing these aspects of the FCDA framework, we briefly describe three prototypes: CIFE

WORLD, AgentCAD, and ACL Project testbed that applied the FCDA framework to different aspects of collaborative facility design and construction planning. Finally, we conclude the paper with a brief summary and conclusions about this research.

2 Design Agents

A design agent is an abstraction for a human designer and a collection of special design software applications called software agents. Human designers are responsible for performing, refining, and critiquing various aspects of design tasks using their software agents. Designers interact with their software agents through user interfaces that are presumed to provide them with rich and powerful communication protocols for exchanging design information with each other. Software agents can be viewed as the designer's assistants in performing some design tasks.

2.1 Software Agents

A software agent represents an engineering or design software application (e.g., CAD, analysis program, etc.) that performs specific design tasks and communicates its design information in the form of notifications and requests using a message-based language called Agent Communication Language (ACL). A notification is an event-driven information message typically generated by a software agent when design information is created, deleted, or updated. For example, when a designer creates, removes, or changes a column using a CAD agent, that would result in a notification message expressing the corresponding action and column design information. A request is a message typically initiated by a software agent on behalf of its user who needs design information. For example, when a designer instructs his/her CAD agent to obtain design information on building spaces, it would generate a request message expressing a query on building spaces. As we will discuss later, other software agents who have information on building design spaces would reply to this type of request.

The activities of a software agent are characterized by interests and capabilities. The interests of a software agent consist of a list of notifications it desires to receive. The capabilities of a software agent consist of a list of requests and tasks it can handle. The interests specify the type and attributes of design objects (elements) as well as the action for notification messages. The capabilities specify the type and attributes of design objects for requests and tasks. Generally, there is an overlap among software agents' interests resulting from designers' shared interests in certain design objects. For example, the interests of a structural design agent in the locations of steel moment resisting frames of a building's structural system overlap with those of an architectural design agent in the building's office spaces. The interests and capabilities are used to route notifications and requests among software agents as will be discussed in Section 3.

2.2 Organization of Software Agents

The organization of software agents is based on the concept of distributed workspaces. Each software agent has an independent workspace that is private to an individual designer to develop design ideas that are not ready to be shared or communicated. Each individual workspace has a unique name across the distributed environment. The design objects in each individual

workspace have an underlying symbolic representation with defined semantics associated with the software agent's specific internal data representation. The symbolic representation consists of a vocabulary that defines design elements as objects with sets of attributes. In performing design tasks, a designer creates and manipulates the internal representation of design objects and the underlying symbolic representation is automatically generated or updated. Each design object is uniquely identified across all workspaces by combining the name of the software agent and a unique local identifier.

Unlike other approaches that use a shared workspace for communications between designers [Fruchter 96b] [Saad and Maher 93], individual workspaces are autonomous, which signifies the independence of software agents from one another in the environment, and are not used as a communication medium. Rather, a networked infrastructure links software agents' distributed workspaces supported by a message-oriented communication model, which provides different modes of interaction between designers based on their information needs and ability to handle certain tasks or information requests. A software agent's workspace captures a specific design vocabulary, its interests in notifications on design objects, and its capabilities in handling specific design tasks as well as certain requests on design objects. Figure 1 shows the distributed workspaces of software agents.

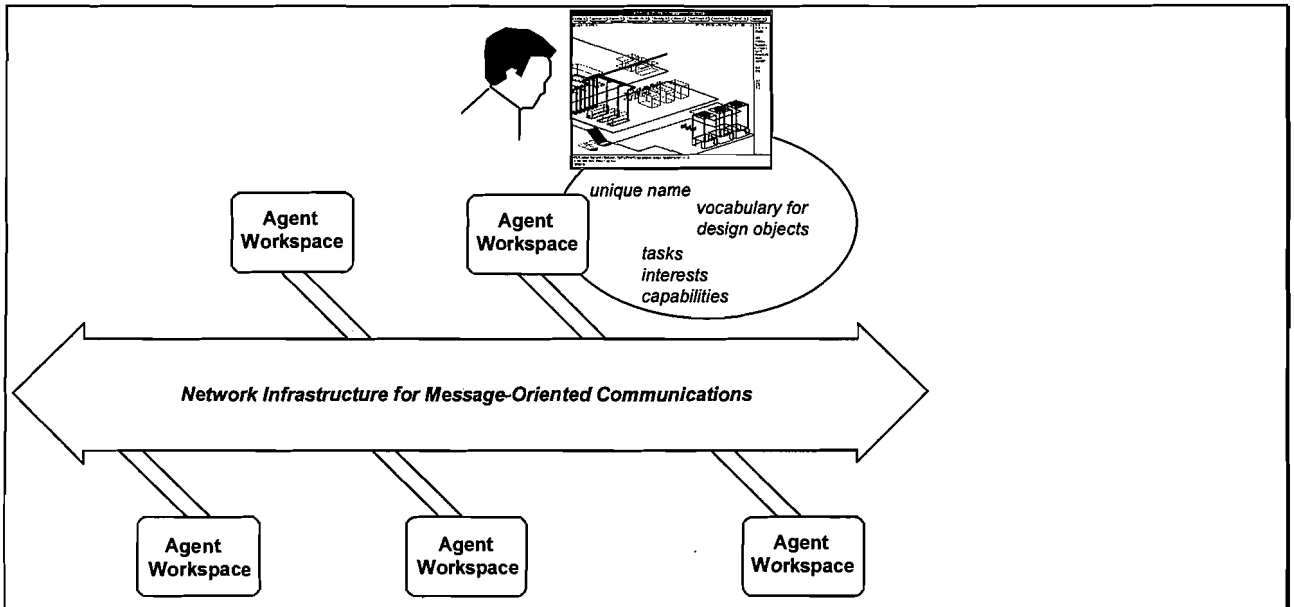


Figure 1: Different agents interact using a message passing across a network.

3 Message-Oriented Communication

The FCDA framework provides for a message-oriented communication which enables distributed software agents to interact via a collection of distributed coordination system programs called Facilitators. These support two styles of communications: an event-driven protocol based on publish/subscribe and a demand-driven protocol based on request/reply using Agent Communication Language (ACL). ACL combines a communication protocol called

Knowledge Query Manipulation Language (KQML) with a declarative logic-based language called Knowledge Interchange Format (KIF) developed by a knowledge sharing initiative [Fikes et al. 91] [Neches et al. 91]. KIF is a prefix version of the language of first-order predicate logic, with various extensions to enhance its expressiveness [Genesereth 92].

Using KIF, various aspects of design information and knowledge can be represented using design vocabularies that describe the geometric data of the physical parts of the product and their relationships, non-geometric information (e.g., details on functionality of the parts, constraints, and design intent), and multiple levels of abstraction. These design vocabularies must have precise semantics and need to be consistent for unambiguous exchange of design information and knowledge.

3.1 Agent Communication Language (ACL)

An ACL message is an expression that starts with the symbol package followed by a list of keyword/value pairs. The general form of a package is as follows:

```
(package    :content <expression>
            :sender <agent-name>
            :receiver <agent-name>
            :reply-with <identifier>
            :in-reply-to <identifier>
            :commode <commode>)
```

The content of an ACL message is an expression typically consisting of a message type and a KIF expression. The message type indicates whether the message is a notification, request, or reply. It is often called a performative based on KQML [Finin and Wiederhold 91]. The KIF expression can be a term, a symbol, or a sentence. Each ACL message must include a notification, request, or reply message specified in the :content field and a sending agent and a receiving agent in the :sender and :receiver fields of the message format. For request messages, the :reply-with field gives the receiver an identifier to use in sending responses to the request. The :in-reply-to field in a reply message is the same identifier of the request to which the current message is a response. Finally, the :commode field indicates the style of reply communication for request messages. The value of this field must be: single, stream, or nil respectively indicating one answer, a stream of answers, or no more answers.

3.1.1 ACL Notification Messages

Software agents communicate design information in the form of notifications when they wish to inform each other of new or updated information resulting from a design task completion or designer's interactive manipulation of design objects. The following is a typical exchange of a notification message on the creation of a column object between agent 3dcad, representing a 3-D CAD system and agent etabs, representing a structural analysis program:

```
(package    :content (tell (column 3dcad-01 2 2 12 (point 10 20 0)))
            :sender 3dcad
            :receiver etabs)
```

In this simple exchange, agent 3dcad has put forward a notification to agent etabs using the term tell, a performative that asserts the truth value of a KIF expression, in this case (column 3dcad-01 2 2 12 (point 10 20 0)). It is imperative that both agents 3dcad and etabs agree on the exact meaning of this performative tell as well as every vocabulary element used, like column, 3dcad-0001, point, and 3dcad-01. For instance, both agents 3dcad and etabs need to agree that the terms column , point, and 3dcad-01 respectively refer to a specific type of column, coordinates of column, and a unique column object identifier. Without this precise agreement, semantic communication cannot take place.

3.1.2 ACL Request/Reply Messages

Software agents communicate in the form of request/reply messages when they need to obtain design information. The following is a typical ACL request/reply exchange in which agent contractor, representing a cost estimation program asks agent pricer, representing a materials price database, for the price of one ton of 4000-psi steel:

```
(package    :content (ask-one (steel-price 4000psi-steel ?p)))
            :sender contractor
            :receiver pricer
            :reply-with 3456)

(package    :content (reply (steel-price 4000psi-steel (dollars 200)))
            :sender pricer
            :receiver contractor
            :in-reply-to 3456)
```

In this example, agent contractor has put forward a request using the term ask-one, a performative that asks for the truth value of a predicate expressed in KIF, in this case (steel-price 4000psi-steel ?p). Agent pricer responds to this request by sending a message using the term reply, a performative specifying the KIF expression that satisfies the request. Both agents contractor and pricer are presumed to agree on the exact meaning of the performatives ask-one and reply as well as every vocabulary element used, like steel-price, dollars, 4000psi-steel, and 200.

3.1.3 Interests and Capabilities

One of the distinctive features of KIF is its ability to represent meta-knowledge. Using this feature, a software agent can specify its interests and capabilities by providing a list of KIF sentences. Here are typical examples of interests and capabilities respectively:

```
(interested etabs '(tell (column ?id ?w ?d ?h (point ?x ?y ?z))))
(handles pricer '(ask-one (steel-price ?c (dollars ?p))))
```

The first sentence declares that agent etabs is interested in the creation of column design information including its dimensions and coordinates. The second sentence declares that agent pricer can provide steel pricing information in dollars. The interests and capabilities of software agents are used by facilitators to enable agents to communicate indirectly.

3.2 A Model for Software Communication

When a designer starts a software agent, it registers its name and declares its interests and capabilities with facilitators. This is called the registration phase. Once a software agent registers with facilitators, it enters its normal operation during which it can communicate with other agents in the form of notifications and requests/replies according to its interests and capabilities.

In this model, software agents acting as clients can issue messages (i.e., notifications and requests/replies) in the system, without regard to their own location in the network or in their host machine. Facilitators are responsible for all mechanisms required to find the other software agents that can fulfill a given message, to forward appropriate messages to those software agents, and to communicate necessary information for making the message. Depending on the type of message (i.e., notification or request) a software agent is issuing, facilitators will either send the notification to interested agents or obtain a reply for the request from capable agents. **Figure 2** illustrates the message-oriented communication model.

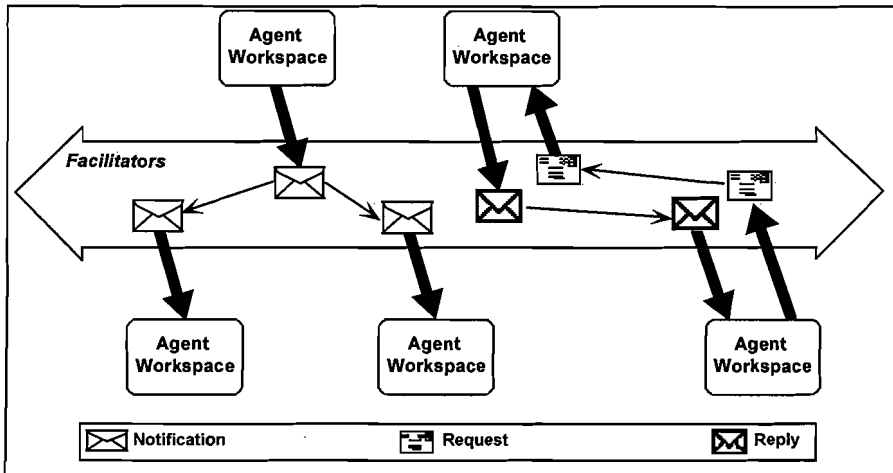


Figure 2: Facilitators send messages to and receive messages from distributed agents.

Facilitators offer a number of services that facilitate the exchange of design information among software agents [Khedro and Genesereth 94b]. They employ logic-based inferential mechanisms to perform these services which include content-based routing of notifications and requests, semantic translation, and request decomposition and compaction. In this section, we discuss these services.

3.2.1 Content-Based Routing of Notifications and Requests

Facilitators perform content-based routing to determine the recipient software agents of communicated notifications and requests. For example, consider the four software agents 3dcad, etabs, contractor, and pricer described in Section 3.1. Suppose that the four agents are now communicating via facilitators, which capture their interests and capabilities described in Section 3.1.3.

Consider the case where facilitators receive from agent 3dcad the following notification indicating the creation of column design information:

```
(package    :content (tell (column 3dcad-01 2 2 12 (point 10 20 0)))
            :sender 3dcad
            :receiver facilitators)
```

In order to determine which software agents are interested in this notification, facilitators form the query (interested ?agent '(tell (column ?id ?w ?d ?h (point ?x ?y ?z)))) and uses their inference mechanisms to find a binding for variable ?agent. In this case, there is just one agent, etabs. Consequently, facilitators send to agent etabs the following notification:

```
(package    :content (tell (column 3dcad-01 2 2 12 (point 10 20 0)))
            :sender facilitators
            :receiver etabs)
```

Now consider the case when facilitators receive from agent contractor the following request for the price of one ton of 4000-psi steel:

```
(package    :content (ask-one (steel-price 4000psi-steel ?p)))
            :sender contractor
            :receiver facilitators
            :reply-with 3456)
```

In order to determine which agents can handle this request, facilitators form the query (handles ?agent '(ask-one (steel-price 4000psi-steel ?p))) and uses their inference mechanisms to find agent pricer as a binding for variable ?agent. Consequently, facilitators forward agent pricer the above request. Upon receiving and processing the above request, agent pricer sends a reply to facilitators, which in turn forward it to agent contractor as follows:

```
(package    :content (reply (steel-price 4000psi-steel (dollars 200)))
            :sender facilitators
            :receiver contractor
            :in-reply-to 3456)
```

3.2.2 Semantic Translation

The second service provided by facilitators is translation, which is the transformation of messages from one form to another. The need for semantic translation arises because of differences between the design product models and abstractions captured by software agents. The semantic translation allows different agents to communicate without being forced to agree on a single product model. As an example of semantic translation, consider agent 2dcad representing a 2-D CAD system and agent concad representing a conceptual CAD system. The interests for both agents 2dcad and concad are expressed as follows:

```
(interested 2dcad '(tell (2dcolumn ?id ?w ?d ?h (2dpoint ?x ?y))))
(interested concad '(tell (concol ?id ?area ?h)))
```

The interests for agents 2dcad and concad in columns are slightly different from the information communicated by agent 3dcad in that agent 2dcad needs the x and y coordinates of a column only while agent concad needs the column's cross sectional area and its height only.

In order to support the communication between agent 3dcad and both agents 2dcad and concad, the following formulation of translation rules are provided in facilitators:

```
(<= (2dcolumn ?id ?w ?d ?h (2dpoint ?x ?y))
        (column ?id ?w ?d ?h (point ?x ?y ?z)))

(<= (concol ?id ?area ?h)
        (column ?id ?w ?d ?h (point ?x ?y ?z))
        (= ?area (* ?w ?d)))
```

Now, suppose that agent 3dcad sends the following notification to facilitators about the creation of a new column:

```
(package    :content (tell (column 3dcad-02 4 4 12 (point 10 20 10)))
            :sender 3dcad
            :receiver facilitators)
```

In this case, facilitators use their inferential mechanisms to translate the KIF sentence (column 3dcad-02 4 4 12 (point 10 20 10)) to the sentences (2dcolumn 3dcad-02 4 4 12 (2dpoint 10 20)) and (concol 3dcad-02 16 12) based on the above rules respectively. Facilitators then check whether any agent is interested in this translated information and find that agent 2dcad is interested in the 2dcolumn sentence while agent concad is interested in the concol sentence. As a result, facilitators send the following two notifications to agents 2dcad and concad respectively:

```
(package    :content (tell (2dcolumn 3dcad-02 4 4 12 (2dpoint 10 20)))
            :sender facilitators
            :receiver 2dcad)

(package    :content (tell (concol 3dcad-02 16 12))
            :sender facilitators
            :receiver concad)
```

3.2.3 Request Decomposition and Reply Compaction (Merging)

Facilitators typically use the capabilities of software agents to identify those agents that can handle incoming requests. In case there is no single agent that can handle an incoming request, facilitators attempt to decompose the request by translating it into a set of equivalent requests that can be handled by a number of agents. Upon receiving replies for the equivalent requests from capable agents, facilitators compact (merge) the replies into a single reply matching the original request and send it to the requesting agent.

To illustrate request decomposition and reply compaction, consider agent euro-contractor, representing a cost estimation program for construction materials that wants to know the price of one ton of 4000-psi steel in a foreign currency, Deutsche Mark. In order to support this request, it would be sufficient to have agent banker that is capable of providing currency exchange rate information as well as a rule that translates the request in Deutsche Marks to another one in Dollars. The capabilities of agent banker and the translation rule are expressed respectively as follows:

```
(handles banker '(ask-one (exchange-rate mark dollar ?r)))

(<= (steel-price ?x (marks ?price))
```

```
(steel-price ?x (dollars ?p))
(exchange-rate mark dollar ?rate)
(= ?price (/ ?p ?rate)))
```

Now suppose facilitators receive from agent euro-contractor the following request inquiring about price of one ton of 4000-psi steel in Deutsche Marks:

```
(package    :content (ask-one (steel-price 4000psi-steel (marks ?p)))
            :sender euro-contractor
            :receiver facilitators
            :reply-with euro-001)
```

Facilitators use (steel-price 4000psi-steel (marks ?p)) and the above translation rule to translate this request into the following two sub-requests that can respectively be handled by agents pricer and banker:

```
(package    :content (ask-one (steel-price 4000psi-steel (dollars ?p)))
            :sender facilitators
            :receiver pricer
            :reply-with facilitators-001)

(package    :content (ask-one (exchange-rate mark dollar ?rate))
            :sender facilitators
            :receiver banker
            :reply-with facilitators-002)
```

Facilitators then simultaneously send these sub-requests to both agents pricer and banker, which provide the following replies respectively about the steel price in dollars and the current exchange rate for the Deutsche Mark:

```
(package    :content (reply (steel-price 4000psi-steel (dollars 200)))
            :sender pricer
            :receiver facilitators
            :in-reply-to facilitators-001)

(package    :content (ask-one (exchange-rate mark dollar 0.65)
            :sender banker
            :receiver facilitators
            :in-reply-to facilitators-002)
```

Upon receiving the two replies from agents pricer and banker, facilitators then combine the two replies, compute the 4000-psi steel price in Deutsche Marks based on the above translation rule, and finally send the following combined reply for the original request initiated by agent euro-contractor:

```
(package    :content (reply (steel-price 4000psi-steel (marks 307.69)))
            :sender facilitators
            :receiver euro-contractor
            :in-reply-to euro-001)
```

3.3 Communication Protocols for Designers

The message-oriented model for software communication enables collaborative interaction among designers in a geographically-dispersed environment. Designers can communicate design information about the tasks they perform as well as access the variety of services through menu-

driven protocols provided by their software agents. The following is a description of the type of collaborative interactions:

- Dynamic Registration and Discovery of Designers: A designer can join or leave a team of designers working on a project by registering or unregistering his/her software agent with the system dynamically. As a part of this registration, the software agent registers its interests, capabilities, and tasks with the facilitators automatically. The designer can also search for the availability of other designers who perform certain design tasks, can handle information requests, and/or are interested in certain design information.
- Communications of Notifications: Designers can communicate information notifications to other designers directly or anonymously without the need to know about their location. An information notification can be the whole current design information, selected design objects, or tracked design changes. In direct communication, the notifications are routed to the specified recipient designer. In anonymous communication, the notifications are routed to those designers whose software agents have registered interest in the information.
- Communications of Requests: Designers can communicate information requests to other designers directly or anonymously. An information request can be a general query about software agent's design objects, a specific query about some attributes of design objects, or a request for a task to be performed. Direct requests are routed to the specified recipient designer while other indirect requests are routed to those designers that can handle them. Received requests are fulfilled, and appropriate replies are forwarded to the requesting designer.
- Communications of Explanations: Designers can communicate explanations and rationale about the design decisions they have made. Those explanations can be communicated directly to a specified designer or indirectly by communicating them under topics or subjects in which other designers express interest.
- Receiving Incoming Messages: Routed incoming notifications, requests, and replies are treated as events by the software agent. Upon receiving notifications, the software agent buffers them automatically and informs the designer of their arrival. Upon the designer's request, these notifications are propagated and displayed in the software agent. Typically, the software agent handles information requests transparently by getting the information that fulfills the request and communicating it in the form of replies.
- Delayed Communications: Designers can choose to be temporarily inactive (off-line), in which case information about their tasks, interests, and capabilities are maintained by the facilitators. For those inactive designers, notifications of interest, tasks and requests handled, and expected replies are temporarily buffered until those designers become active (on-line) again, in which case the buffered messages are communicated.

4 Modeling Design Vocabularies for Communications

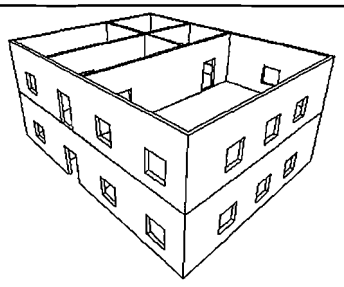
Modeling design vocabularies is a critical step to supporting unambiguous communications among software agents. In the FCDA framework, KIF is used to formally formulate and express design vocabularies used in ACL message-based communication. The modeling approach is to

develop vocabularies based on discipline models captured by the different applications and formulate relationships that describe semantic relationships as translation rules between relevant terms of each discipline vocabulary. The FCDA modeling approach is significantly different from database centric approaches which aim at developing a common globally-consistent product model shared by software applications.

In building design and construction modeling, the vocabularies typically have as their foundation the appropriate models of architectural schematic design, structural design, and construction planning. Each of these vocabularies describes the components and appropriate relationships for each of the discipline models. For example, the structural design vocabulary describes the geometry, topology, and behavior of the structural system's components (e.g., columns, girders, floor panels, sub-beams, and frames) and their relationships. Different subsets of this vocabulary are shared by different discipline software agents. This section briefly discusses the vocabularies of the models that correspond to various architectural and structural design as well as construction planning tasks of the conceptual design of steel buildings.

4.1 Architectural Design Vocabulary

At the conceptual stage, an architectural design model expresses the spatial requirements for the form and function of the building. This model typically describes a schematic design (block diagram) that specifies desirable spaces, their relationships to each other, the number of stories, and necessary openings, windows and doors for the intended building. The primary dimensions of the objects in the schematic design, their width, depth, and height, are determined respectively in three principal orthogonal directions x, y, and z of a global reference frame. Figure 3 shows an example of a 3-D architectural schematic design and sample vocabularies for a wall expressed in KIF.



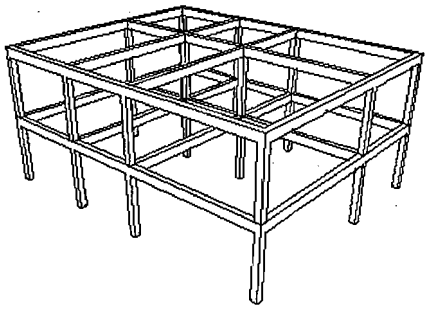
```
(wall waj*)
(= (wall.part-of waj*) abld*)
(= (wall.type waj*) load-bearing)
(= (wall.story waj*) 1)
(= (wall.length waj*) 100)
(= (wall.depth waj*) 2)
(= (wall.height waj*) 15)
(= (wall.coordinates waj* global*) (list 0 40 0 0))
```

Figure 3: The Facilitator project specified a vocabulary for architectural design. The right side of this figure shows how KIF expressions define some of the attributes and attribute values of the space shown in the left side of the figure.

4.2 Structural Design Vocabulary

The structural design model supports spaces and functions intended for the building. It describes a structural system that will safely transfer to the ground building loads that arise from gravity, earthquakes, and wind. A steel structural system is a collection of steel structural components interconnected to transfer and distribute different types of loads the building has to resist. The

behavior of a structural system as a whole is defined in terms of the behavior of its constituent structural components. The behavior of the structural system describes the behavior of the building. Generally, the primary structural components of steel structures, which resist vertical service loads, are columns, girders, and floor panels (a set of parallel steel sub-beams topped by a reinforced concrete deck). A steel structural component is defined by a set of attributes that describe its geometry, its topology, and its behavioral characteristics under applied external loads. In addition to the primary components, there are other important structural components, which resist horizontal loads resulting from wind and seismic activities. These components are moment-resisting frames, braces, and shear walls, and they can be represented similarly. Figure 4 shows an example of a 3-D steel structural system and a sample representation for a floor panel expressed in KIF.



Sample floor panel vocabulary

```
(floorpanel flpj*)
(= (floorpanel.part-of flpj*) sbld*)
(= (floorpanel.floor flpj*) 2)
(= (floorpanel.length flpj*) 100)
(= (floorpanel.width flpj*) 40)
(= (floorpanel.thickness flpj*) 1)
(= (floorpanel.coordinates flpj* global*) (list 50 20 15))
(= (floorpanel.subeams flpj*) (list sb1* ... sbn*))
(= (floorpanel.deck flpj*) rcdj*)
(= (floorpanel.dist-load flpj*) 100)
```

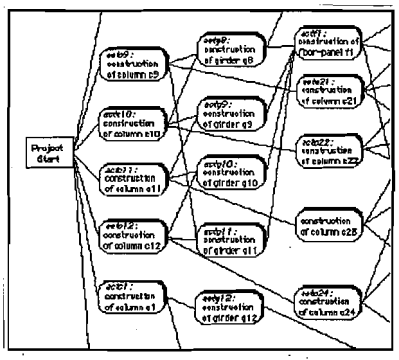
Figure 4: The right side of this figure shows how KIF expressions define some of the attributes and attribute values of the structural system shown in the left side of the figure.

4.3 Construction Planning Vocabulary

The construction plan describes the activities necessary for building such a structural system, which represents the main skeleton of a building. The vocabulary of a construction plan for a structural system involves describing a collection of activities necessary for the construction of this system (for simplicity we exclude the ancillary activities for the procurement and delivery of the structural elements). These activities are connected by a number of different types of relationships. One kind of relationship involves sequential dependency, that is, before or after dependency, which reflects the ordering among activities.

A primary activity involves the construction of a structural component. It is based on the construction materials, type of structural component, and the relationship of the primary activity to other activities. Secondary activities involve specific tasks (e.g., placing girders, connecting columns, pouring concrete, etc.) and are typically considered as sub-activities of the primary activities. In the model presented, secondary activities are considered to be the actual construction operations whose execution is based on the anticipated equipment and personnel allocations for a particular project. The execution of each construction operation requires specific equipment and special personnel for some period of time. A series of construction operations (secondary activities) comprises a primary activity. Figure 5 shows an example of a high-level construction plan that illustrates the precedence of activities involved in the

construction of various structural components (e.g., columns, girders, floor panels, etc.) and a sample construction activity vocabulary expressed in KIF.



Sample construction activity vocabulary

```
(construction-activity actk*)
(= (activity.type actk*) primary)
(= (activity.component actk*) gdrk*)
(= (activity.predecessor actk*) act1*)
(= (activity.predecessor actk*) actm*)
(= (activity.successor actk*) actn*)
(= (activity.operations atck*) (list cop1* cop2* cop3*))
(construction-operation cop3*)
(= (operation.type cop3*) erection)
(= (operation.required-equipment cop3*)
                                (list crane power-ridges))
(= (operation.period cop3*) 10)
```

Figure 5: The right side of this figure shows how KIF expressions define some of the attributes and attribute values of construction activities, shown in a network in the left side of the figure.

4.4 Interdisciplinary Translation between Design Vocabularies

In order to bridge the communication gap between applications which use different vocabularies, translation rules are developed to semantically translate between the relevant terms of each discipline's vocabulary. The translation rules are expressed as a collection of KIF axioms and provided in the facilitators. The rules for translation between the three vocabularies are crafted in a way that maintains the consistency of communicated information between the three disciplines. The purpose is neither to unify the different product models nor to achieve a comprehensive consistency among them.

As an example of interdisciplinary translation, the following is a KIF axiom that expresses the relationship between the coordinates of a load-bearing wall within an architectural schematic design and the coordinates of a moment-resisting frame within the structural system:

```
(<== (= (wall.coordinates ?w ?rf) (list ?x ?y ?z ?a))
        (= (wall.type ?w) load-bearing)
        (frame-wall ?f ?w)
        (= (frame.type ?f) moment-resisting)
        (= (frame.coordinates ?f ?rf) (list ?x ?y ?z ?a)))
```

If ?w is a load-bearing wall that corresponds to the moment-resisting frame ?f, whose coordinates are (list ?x ?y ?z ?a), then the coordinates of the wall are (list ?x ?y ?z ?a).

5 Federated Agent/Facilitator Architecture

5.1 General

As with other broker services, messages from agents to their facilitators are undirected in that they only have content but no addresses. It is the responsibility of the facilitators, as shown in

Figure 6, to route notification messages to those agents that are interested, and request messages to the agents that are able to handle them. The architecture is built on a reliable TCP/IP communication layer providing the basic protocols needed to connect and communicate with software agents throughout the underlying network and different operating systems. Each facilitator consists of a number of procedures for communication and message processing. Engineering applications are interfaced with Agent Programmatic Interfaces (APIs) to adapt them to behave as software agents and to communicate with facilitators. This section discusses the facilitator's architecture as well as interfacing engineering software.

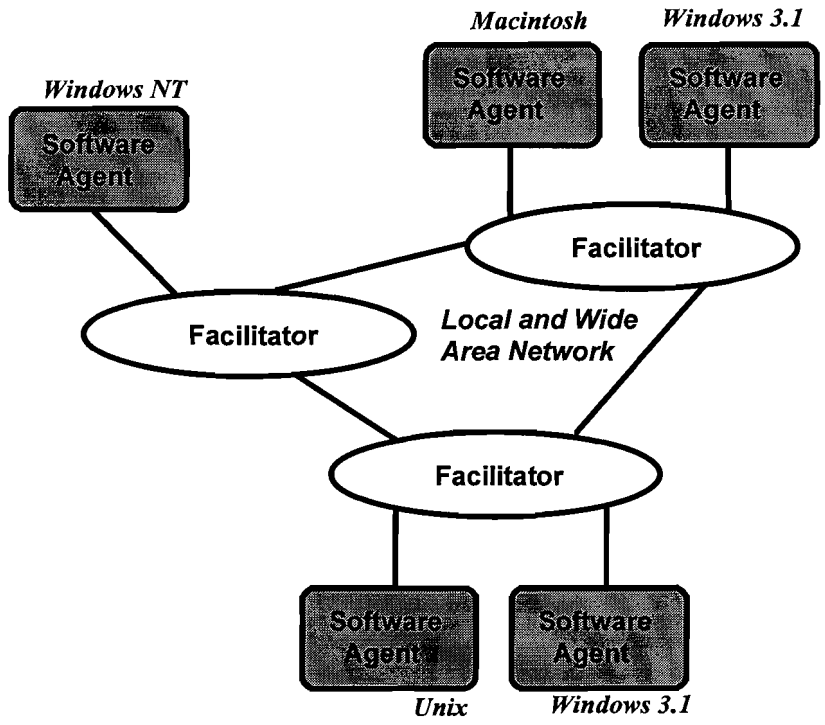


Figure 6: The FCDA framework provides a federated distributed system architecture, in which each group of software agents communicate only with a single facilitator, and facilitators communicate with each other.

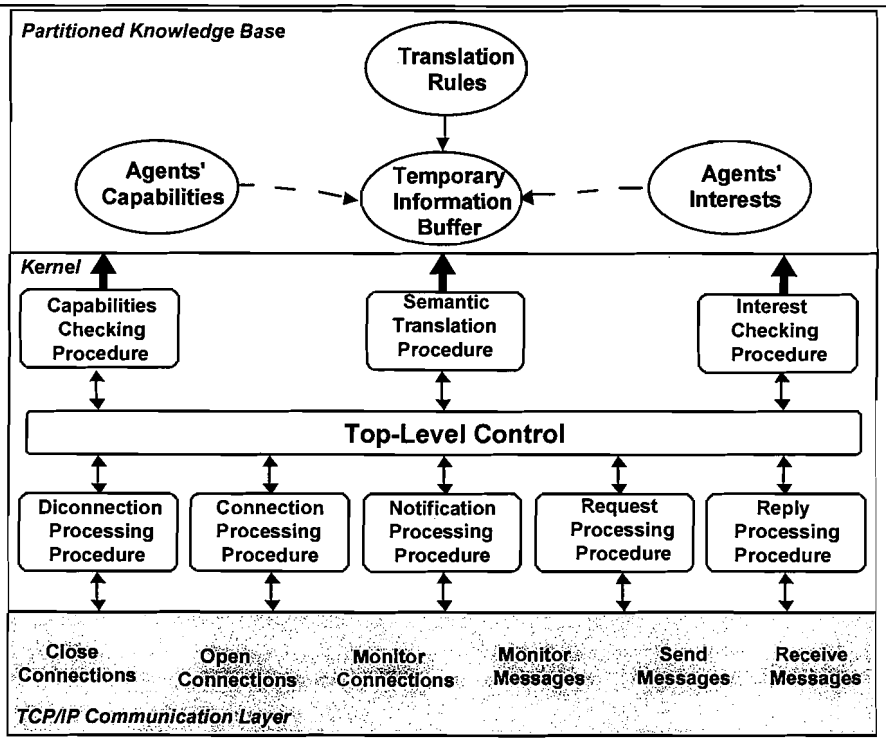


Figure 7: Each facilitator has a layered architecture that includes low-level message-handling support for network communication, a kernel that provides message control, and a high-level set of knowledge bases that specify agents' interests, capabilities, and inter-disciplinary translation rules.

5.2 Facilitator's Architecture

The facilitator architecture consists of several communication procedures and automated inference procedures operating on a partitioned knowledge base. The communication procedures operate on the TCP/IP layer for message delivery and monitoring, agent location, and agent connections and disconnections. The facilitator's communication procedures perform various functions for processing and managing communication links with software agents and for processing agents' notifications, requests, and replies. This processing is orchestrated by a top-level control procedure, which also executes the other three procedures responsible for performing automated inference on the partitioned knowledge base. The dashed arrows among these partitions in the knowledge base indicate temporary inclusions that are used for controlled inference. The inference procedures include backward chaining and forward chaining as well as a hybrid form of both. A facilitator uses its knowledge base to perform the services of: content-based routing of notifications and requests, semantic translation, and request decomposition and reply compaction.

5.3 Interfacing Engineering Applications

The FCDA framework provides APIs that standardize adapting engineering software to be software agents as well as for developing new engineering applications as software agents. The API is a subroutine library that developers can use to interface their applications. It includes a number of procedures that facilitate communications of software programs in the form of ACL messages. The API hides the boundaries of various system dependencies including network configuration from developers. Using an API, software applications maintain different views of the tasks they perform as well as their own internal representations of the communicated design information. The API bi-directionally translates software agents' representations into equivalent KIF representations. In this way, it enables software applications to capture communicated ACL messages in their own internal representations. At the same time, they inherit the agent's behavior. The API is implemented using different programming languages and software applications including C/C++, Lisp, AutoCAD, and Microstation to support different types of engineering applications.

6 Experiments in Collaborative Distributed Facility Engineering

Using the FCDA framework, we have developed three collaborative distributed environments. The first one, called CIFE WORLD, focused on integrating eight different types of software applications that spanned architectural design, structural design, and construction planning of buildings [Khedro 94a]. The second prototype, called AgentCAD, focused on the use of CAD to examine aspects of designers' collaboration in a geographically-dispersed environment [Khedro 96]. The third development was a multi-institutional project called Agent Collaboration Language Project (ACL Project) [Khedro et al. 95]. The project aimed at developing a testbed in cooperation with other four universities to integrate independently-developed design and engineering systems over the Internet. This section describes these environments.

6.1 CIFE WORLD: Collaborative Distributed Building Engineering Environment

CIFE WORLD is a collaborative environment developed using the FCDA framework. It consists of four design applications that are used by an architect, a structural designer, and a foundation engineer as well as management applications used by a construction engineer and a contractor for construction planning, scheduling, and cost estimation of a building. The applications comprise two extensions of AutoCAD for architectural and structural design (ArchCAD and StrucCAD respectively), a structural design system (BERT), a footing design program (FOOTER), a building cost estimating program (Estimator), a knowledge-based construction planning system (OARPLAN), a construction scheduling system (Scheduler), and an ORACLE DBMS [Khedro 94a]. CIFE WORLD's architecture is depicted in Figure 8. For CIFE WORLD, we have developed three vocabularies for architectural design, structural design, and construction management with the necessary translation rules. Each application has been interfaced with the appropriate implementation of API by specifying the communication vocabularies. The eight software applications have been supported by two facilitators through an Internet-based network

of seven different computers running on various types of hardware and software, which includes Sun workstations, DEC workstations, Macintoshes, and PC's.

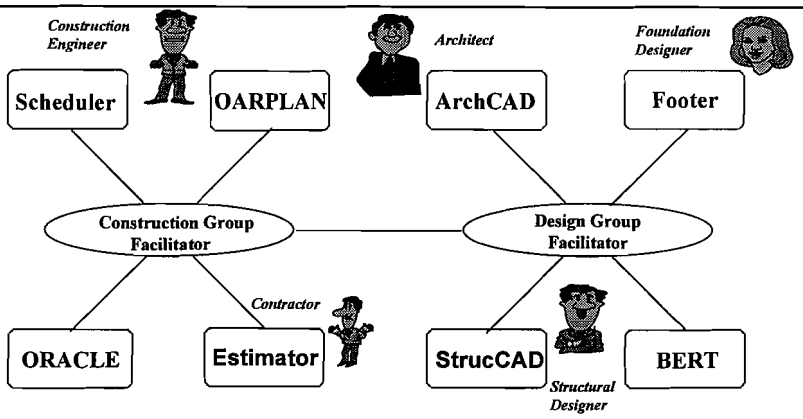


Figure 8: The CIFE WORLD demonstration project included a set of different applications on different computers that link to each other across a network using facilitators.

6.1.1 Evolution of Concurrent Building Design and Engineering Tasks

In the CIFE WORLD environment, the process of building design and construction starts when the architect performs the architectural schematic design using ArchCAD based on the design requirements. Once the architect communicates the design information, the structural designer receives general information about the schematic design through BERT. Using BERT, the structural designer designs the configuration of a building's structural system and selects design sections for all its structural components based on recommended structural design loads and specific steel design properties. When the structural designer communicates this information, appropriate design information is simultaneously forwarded to: StrucCAD, FOOTER, Estimator, and OARPLAN.

Using StrucCAD, the structural designer can view the geometry of structural system design in 3-D. Meanwhile, the foundation engineer receives appropriate information about the first-story columns, including the loads they are resisting. Using FOOTER, the foundation engineer designs reinforced concrete footings taking into account the building site's soil properties and communicates the design information. At the same time, the construction engineer receives information about the relationships among various structural components in OARPLAN, produces a construction plan including various activities, and communicates the information about the activities.

Once Scheduler receives the activity information, the construction engineer generates a schedule of various construction operations taking into account the anticipated personnel and equipment allocation. In parallel with the footing design and construction planning and scheduling tasks, the contractor, using Estimator, receives the necessary information about the structural component design sections. After obtaining the labor and material unit costs from the unit cost database, he/she produces a conceptual cost estimate for the structural system. When the footing

design information is communicated by the foundation engineer, both the construction engineer and contractor receive the updated information. As a result, the contractor produces a final conceptual cost estimate and the construction engineer generates a final plan and schedule.

6.1.2 Communication of Design Change and Its Implication

In CIFE WORLD, when a project participant communicates a design change, the implications of this change are determined and forwarded to the affected applications. For example, say the structural designer decides to move an internal moment-resisting frame a distance of 5 feet in some direction within BERT. When he/she communicates this design change, all relevant, changes including those bearing walls and doors that are affected by the moved frame, are determined and forwarded to ArchCAD, StrucCAD, FOOTER, and Estimator. Note that since moving a frame does not change the physical relationships among structural components, OARPLAN is not informed about any changes.

After receiving the appropriate changes for footing locations and first-story column design loads, the foundation engineer, using FOOTER, re-designs footings by selecting new reinforcement specifications while maintaining the same footing dimensions. The designer then communicates these changes, which are forwarded to Estimator for a revised cost estimate to be produced.

6.2 AgentCAD: Distributed Collaborative CAD Environment

AgentCAD is a network infrastructure of distributed CAD applications that facilitates the concurrent and collaborative interaction of several designers working together, possibly over several physical locations, on a design project [Khedro 96]. It provides a set of services and protocols that support the communications of distributed design information captured by CAD drawings, multiple design views, and design changes. It coordinates access to a common and multiple design models as well as the activities of designers based on captured knowledge of designers' tasks, capabilities, and interests, which characterize their behaviors.

6.2.1 Coordination of Information Exchange

AgentCAD environment provides distributed CAD workspaces to a team of designers performing different design tasks of a common design project. AgentCAD coordinates the concurrent and asynchronous communications among them by enforcing ownership on the design objects being created and communicated. In the AgentCAD environment, designers communicate about the design objects they create, remove, or change. When a designer creates a design object within his/her workspace the object's unique identifier, which includes the name of the workspace, is communicated along with the object information. As a result, any recipient CAD workspace of this design object can determine its creator. With this information available at every CAD workspace that registered interest in this design object, no CAD application allows its user to remove or change the design object if it is not the owner of that object. This approach enforces some control on how changes to existing design objects are made and communicated in a distributed environment. With this level of control, conflicting changes are avoided. In order to enable designers to make or propose changes to design objects created by other designers,

AgentCAD provides a protocol for ownership transfer of design objects from one workspace to another. In transferring ownership, the object's unique identifier is changed to include the name of the workspace, which owns the object. This change is propagated to all other workspaces.

6.2.3 Illustrative Scenario for Interaction of Designers

We illustrate the use of AgentCAD using a design test case of a hospital's equipment building [Collier and Fischer 95]. For this test case, an architect, a structural designer, and a mechanical engineer are cooperating on various aspects of architectural, structural, and HVAC design. Figure 7 shows the three perspectives of the building model. These perspectives represent the architectural, structural, and HVAC design disciplines as viewed by the architect, the structural designer, and the mechanical engineer respectively. In this design scenario, the mechanical engineer plays an important role in specifying the space requirements to accommodate the various types of HVAC equipment. This equipment includes boilers, cooling towers, electrical equipment, pumps, and pipes that support the hospital's functions. Figure 7a shows a close-up view of some of the equipment as viewed by the mechanical engineer. The structural designer's role in this design is primarily to design for the building's foundation with a special consideration to the weight of the heavy equipment (e.g., cooling towers). In addition, the structural designer is responsible for designing the steel structural frame and the building's reinforced concrete roof. Figure 7b shows a 3-D view of the equipment building as viewed by the structural designer. The architect is concerned with the adequacy of spaces, circulation, and aesthetics of the building to support its functions. In addition, the architect in this scenario is responsible for identifying any design conflicts that may arise from the limited design views which the structural designer and mechanical engineer may have. Figure 7c shows a 2-D view of the equipment building as viewed by the architect.

The different tasks of the current hospital design have been performed independently and design information was communicated using the AgentCAD distributed environment. In this process, facilitators route the appropriate created design objects and changes produced by one designer to the others. For example, if one designer moves pipes and pumps within his/her CAD application and communicates these changes, the other designer would be informed of these changes if interested. These changes can be propagated dynamically and reflected in the drawing of the other application.

6.3 Agent Collaboration Language Project (ACL Project)

Agent Collaboration Language Project (ACL Project) is a multi-institutional project that involved a team of researchers representing four major universities: Carnegie Mellon University (CMU), Massachusetts Institute of Technology (MIT), Stanford University (Stanford), and the University of Illinois at Urbana-Champaign (UIUC) under the Architect Associate program of Construction Engineering Research Laboratory of US Army Corps of Engineers (USACERL). The ACL Project involved the development of a multi-institutional testbed for a collaborative facility engineering infrastructure [Khedro et al. 95]. The objectives were to examine aspects of facility design modeling and interdisciplinary semantic translation, and to demonstrate

collaborative design of a simple cabin over the Internet. The FCDA framework served as the basis for integrating these systems in a federated agent/facilitator architecture.

6.3.1 Description of Testbed

The testbed included an architectural design system called SEED developed by CMU, a structural engineering environment called StruDA developed by MIT, an energy analysis environment called BLAST developed by UIUC, and Owner and Project manager agents implemented in ACE environment developed by USACERL. Figure 8 depicts the architecture for the testbed.

- SEED: a software environment intended to support the early phases in building design, aims at providing computational support for the generation of an architectural program, the schematic layout of the functional units of the program, and the generation of a 3-D configuration of spatial and physical building components from schematic layout [Flemming et al. 94]. SEED starts from a rough project specification (e.g., building type, size, budget, site description, and other context-specific information). The architectural program generated with the assistance of SEED consists of the basic functional units needed for a building with the specified characteristics and specifies requirements (shape, performance) that a configuration of these units must satisfy. A major component of the SEED environment is the SEED-Layout module, which provides support for the schematic layout phase in building design.

- StruDA: an object-oriented system for designing structural systems. It provides a powerful design knowledge representation scheme, maintains design alternatives and context information, and allows visualization of alternatives [Chiou 96]. StruDA supports design as a synthesis process, involving an arrangement of pre-defined building blocks to compose the design. While such a synthesis is primarily an aspect of routine design, the flexibility of representation and a fine granularity of the building blocks allow for potentially innovative design solutions to be materialized.

- BLAST: Building Loads Analysis and System Thermodynamics program is a comprehensive computer program used for predicting energy consumption and energy systems performance in buildings [BLAST 91]. It conducts an analysis of the thermal behavior of the building and provides information to improve comfort and reduce life cycle cost. BLAST needs a description of the preliminary design of the building geometry, heating and cooling loads, and other useful information for thermal simulation.

- ACE: Agent Collaboration Environment (ACE) is an implementation of a general purpose model of agent collaboration called the Discourse Model [Case and Lu 95]. Simply put, ACE treats assertions by agents as opinions, and provides functionality for agents to discuss those opinions. Project Owner and Project Manager agents have been implemented in ACE. Their purpose is to establish the initial functional requirements for a facility, and situate the facility on a site. They also provide visualization capabilities through ACE's interface with commercial CAD software.

6.3.2 Object and Vocabulary Modeling

In the ACL Project, agents used an object-centered representation of product models that is rich in content and highly structured with complex relationships between objects. Agents expressed their own representation relevant to the tasks they performed using Object Modeling Language (OML), which supports classes, single and multiple inheritance, relationships, and other concepts [Snyder and Flemming 94]. Despite the fact that all the agents used the OML modeling language, their underlying object-oriented representations and implementations were significantly different.

Agents were interfaced with the appropriate APIs, which bi-directionally translated their internal representations into equivalent KIF representations and enabled them to communicate in the form of ACL messages with a single facilitator. To semantically translate between the diverse perspectives used by different agents, the facilitator was provided with a collection of KIF rules that expressed the semantic relationships between the different concepts captured by agents' object models. In addition, the facilitator was provided with the set of agents' interests and capabilities that are respectively used in routing notifications and requests.

6.3.3 Formulation of Translation Rules

In the ACL Project, we have formulated over 500 translation rules to translate between the four distinct design models captured by ACE (Owner and Manager), SEED, StruDA, and BLAST. In order to prevent conflict between the underlying semantics of different vocabularies, i.e., class names, slot names, and relationship names, corresponding to the four different models, we have used four different unique name spaces that provided contexts for the different design vocabularies.

The complexity of the translation among the four design models varied depending on how different the representations were. Essentially, the rules we have formulated can be classified into the following three categories:

- One-to-one simple translation: this type of translation rule was formulated for the cases in which agents used different vocabulary terms, but fundamentally agreed on the underlying semantics and characteristics of an object including class names and slot names. An example of this includes StruDA's Story class name and SEED's Storey class name.
- One-to-many decompositional translation: this type of translation rule was formulated for the cases in which an object in one agent's model is represented by several objects in another agent's model. An example of this is ACE's Site class translated to SEED's Site and Rectangle classes, with a relation expressing the relationship between them. With this type of translation rule, new object identifiers may be created when an object creation message is translated to multiple object creation messages.

- Many-to-one compositional translation: this type of translation rule was formulated for the cases in which several objects in one agent's model are translated to represent one object in another agent's model. An example of this is the translation of StruDA's StructSystem objects (e.g., Beam, Column, etc.) into the equivalent SEED objects. The StruDA object was represented by an object that denoted location, and another object that denoted the structural component's physical characteristics such as size, orientation, etc. In SEED, the equivalent structural component object had attributes that described both the position and location. This problem was handled by buffering partial information. When StruDA communicated a StructSystem object, its attributes were buffered until a GeometryLoc object was also sent. At that point, the two objects were combined, and the results were passed along to SEED. Thus, there was considerable effort to both understand the design context and representation for the objects in each environment.

7 Summary and Conclusions

In developing CIFE WORLD, AgentCAD, and ACL Project, we have examined how the aspects of FCDA framework can support the integration of engineering software and the collaboration among their users. The concept of software agents extends the ways engineering software communicate design information from a limited file-based data exchange to message-based notifications and requests/replies of semantic information. Software agents can communicate design information and changes partially and incrementally. This enables efficient communications compared to a batch file-based exchange. Software agents capture the designers' information needs and tasks, which support designers' collaboration through enabling them to gain understanding of each other's design activities.

The message-oriented communication provides a flexible model for designers to exchange, access, and obtain necessary design information across the network without the need to know where the information resides. In addition, the model supports performing concurrent design tasks among distributed designers as demonstrated in both CIFE WORLD and AgentCAD environments. Compared with query-based communications commonly supported by DBMS, the message-oriented model provides richer communication protocols that better support the collaboration of designers. This includes one-to-many routing of notifications and requests that enable designers to notify and obtain any design information to and from other designers.

The use of KIF in modeling design vocabulary offers expressive language for describing various aspects of architectural and structural design, energy analysis, and construction planning. The vocabulary modeling includes expressing multi-disciplinary product models as well as inter-disciplinary semantic translation rules among them. Overall, this modeling approach enables agents to communicate information according to their models and does not force them to share one product model. It relies on the inter-disciplinary translation rules spanning different product models to bridge the communications between agents.

The advantage of this approach is the ability to evolve (i.e., extend) the model without impacting other agents in the environment. However, the formulation and maintenance of translation rules can be quite challenging in the case of complex product models. The benefits for model evolution were observed in the development of CIFE WORLD where there was a need to extend both the structural design and construction planning vocabularies to support the communication about the behavior of structural columns as well as activity information. In this extension, we had to add vocabulary terms and did not need to change any of those that were already defined.

In the case of the ACL Project, however, we faced a major challenge in the formulation of translation rules. The challenge arose from the fact that agents used four distinct discipline product models that used complex object-oriented representations and were created by different teams. For this reason, the rule formulation was not easy to accomplish since we had to consult with the other project participants frequently to verify the underlying semantics of each model. From a translation standpoint, the ACL Project was an extreme case since there were some similarities among the four models' underlying semantics from which a subset common vocabulary could have been derived. Common product models among agents certainly help reduce the translation effort and should be encouraged whenever possible. Semantic translation can play a significant role in bridging inherent differences among parts of product models where a unified model is difficult to achieve.

The federated agent/facilitator architecture offers adequate flexibility in integrating existing engineering applications over any network including the Internet. Through agent APIs, it accommodates existing engineering applications running on different platforms. In the ACL Project, participants from other universities were able to integrate their applications by interfacing them with a common API, which enabled their communications with a single facilitator in the form of ACL messages over the Internet. One of the key features of this architecture is its ability to integrate applications incrementally. For example, the CIFE WORLD environment has been developed in two stages. Initially, only four applications ArchCAD, BERT, StrucCAD, and OARPLAN had been integrated around a single facilitator. The other four applications were then incrementally integrated, forming the current system configuration based on two facilitators. Since the communication between CIFE WORLD's applications are independent of the configuration of these applications, it enables the applications to communicate with two facilitators without any further changes. This was an important observed flexibility of the federated agent/facilitator architecture. Compared with a blackboard architecture that employs a monolithic central global database, the federated agent/facilitator architecture distributes information and control among agents. Another observation was that the federated agent architecture preserves the autonomy of the applications and thereby provides for more concurrency in performing design and construction activities. However, the consistency of information distributed among the design and engineering software is at risk because design changes can be made and communicated concurrently. We have partly addressed this problem in the AgentCAD environment by controlling who can make certain changes through object/information ownership.

The FCDA framework represents a new direction for the integration of engineering applications. The framework has primarily focused on the modeling, communication, and coordination of design information as well as application integration. These are critical to supporting collaborative design in a distributed environment. Other issues that have not been addressed by the framework and are equally important include: management of design versions, identification and resolution of design conflicts and capturing and communicating design intent. In our future research efforts, we will try to address the conflict issue by developing collaboration strategies that support the negotiation of design alternatives. We hope to benefit from current on-going experiments conducted as part of a project-oriented course which introduces civil engineering graduate students to emerging computer collaboration technologies and concepts of cooperative teamwork [Fruchter 96a]. The AgentCAD environment is used by the students to work on joint design projects between the Stanford and UC Berkeley campuses.

Acknowledgments

The authors wish to acknowledge the significant contributions of Michael Genesereth in supporting and supervising Taha Khedro's Ph.D. research work, which led to the development and application of the FCDA framework. The authors also wish to acknowledge the contributions of many participants in the ACL Project, most notably Michael Case, Jen-diann Chiou, Ulrich Flemming, Tun Gyaw, Eric Leveque, Bob Logcher, Dan Malmer, Curtis Pedersen, Maria Pflaum, Tony Rondonuwu, James Snyder, Ram D. Sriram, Philip Tai, and Murali Vemula. This research has been supported by the Center for Integrated Facility Engineering at Stanford University and the US Army Construction Engineering Research Laboratory (USACERL).

References

[BLAST 91] BLAST Support Office (1991) BLAST User's Reference Vol. I and II, Dept. of Mechanical and Industrial Engineering, Univ. of Illinois at Urbana-Champaign.

[Case and Lu 95] Case, M. P. and Lu, S. C-Y. (1995) A Discourse Model for Collaborative Design. Journal of Computer Aided Design.

[Chiou 96] Chiou, J. (1996) Testing a Federation Architecture in Collaborative Design Process. Master's Thesis, Dept. of Civil and Environmental Engineering, MIT.

[Collier and Fischer 95] Collier, E. and Fischer, M. (1995) Four-Dimensional Modeling in Design and Construction. Technical Report, Center for Integrated Facility Engineering, Dept. of Civil Engineering, Stanford University.

[Fikes et al. 91] Fikes, R., Cutkosky, M. , Gruber, T., and Van Baalen, J. (1991) Knowledge Sharing Technology. Research Report KSL91-71, Knowledge Systems Laboratory, Stanford University.

[Finin and Wiederhold 91] Finin, T. and Wiederhold, G. (1991) An Overview of KQML: A Knowledge Query Manipulation Language. Research Report, Dept. of Computer Science, Stanford University.

[Flemming et al. 94] Flemming, U., Coyne, R., Fenves, S., Garrett, J., and Woodbury, R. (1994) SEED: A software environment to support the early phases in building design. Proceedings of Internationales Kolloquium ueber Anwendungen der Informatik und Mathematik in Architektur und Bauwesen, Weimar, Germany.

[Fruchter 96a] Fruchter, R. (1996a) Multi-Site Cross-Disciplinary A/E/C Project Based Learning. Proceedings of the Third Congress of Computing in Civil Engineering, Anaheim, CA.

[Fruchter 96b] Fruchter, R. (1996b) Conceptual Collaborative Building Design Through Shared Graphics. IEEE Expert Journal, 3(11) 33-41.

[Genesereth 91] Genesereth, M. R. (1991) Designworld, Proceedings of IEEE Conference on Robotics and Automation.

[Genesereth 92] Genesereth, M. R. and Fikes, R. E. (1992) Knowledge Interchange Format. Report Logic-92-1, Dept. of Computer Science, Stanford University.

[Genesereth 94] Genesereth, M. R. and Katchpel, S. P. (1994) Software Agents. Communications of the ACM, 37(7) 48-53.

[Khedro 94a] Khedro, T. (1994a) A Distributed Problem Solving Approach to Collaborative Facility Engineering Through Agent-Based Software Integration. Ph.D. Dissertation, Dept. of Civil Engineering, Stanford University.

[Khedro and Genesereth 94b] Khedro, T. and Genesereth, M. R. (1994b) The Federation Architecture for Interoperable Agent-Based Concurrent Engineering Systems. International Journal on Concurrent Engineering, Research and Applications, 2: 125-131.

[Khedro et al. 95] Khedro T., Case, M. P. , Flemming, U., Genesereth, M. R., Logcher, R., Pedersen, C., Snyder, J., Sriram, R. D., and Teicholz, P. M. (1995) Development of Multi-Institutional Testbed for Collaborative Facility Engineering Infrastructure. Proceedings of the Second Congress on Computing in Civil Engineering, Atlanta, GA.

[Khedro 96] Khedro, T. (1996) AgentCAD: A Distributed Cooperative CAD Environment, Proceedings of International Conference on Information Technology in Civil and Structural Engineering - Taking Stock and Future Directions, Glasgow, Scotland, UK.

[Neches et al. 91] Neches, R., Fikes, R. , Finin, T. , Gruber, T., Patil, R. , Senator, T., and Swartout, W. (1991) Enabling Technology for Knowledge Sharing. *Artificial Intelligence Magazine*. 12:36-56.

[Saad and Maher 93] Saad, M. and Maher, M. L. (1993) A Computational Model for Synchronous Collaborative Design, Workshop on Artificial Intelligence in Collaborative Design, AAAI Press.

[Snyder and Flemming 94] Snyder, J. and Flemming, U. (1994) ACL Object Model Specification Language. Technical Report, Dept. of Architecture, Carnegie Mellon University, Pittsburgh, PA.