A development framework for data models for computer-integrated facilities management

Kevin Yu a,b,1, Thomas Froese a,*, Francois Grobler c,2

a Department of Civil Engineering, University of British Columbia, 2324 Main Mall, Vancouver, BC, Canada V6T 1Z4
b Timberline Software Corp., 15195 NW Greenbrier Parkway, Beaverton, Oregon, USA
c US Army Corps of Engineers, Construction Engineering Research Laboratories, Engineering Processes Division, USA

Abstract

Open computer-integrated facilities management systems hold the promise to improve facilities management practice, but they require extensive underlying technical foundations: particularly standardized data models to enable information sharing among computer applications. The International Alliance for Interoperability is developing Industry Foundation Classes to provide this type of support for all architecture, engineering, construction, and facilities management industries. Facilities Management Classes are a similar effort in advance of, and in extension to, the Industry Foundation Classes for facilities management. This paper presents a framework for the development of Facilities Management Classes and computer-integrated facilities management systems, including objectives, methodologies, implementation issues, etc. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Computer integrated facilities management; Data modeling; Facilities management; Industry foundation classes; International alliance for interoperability

1. Introduction

In any architecture, engineering, and construction (AEC) activity, information is created by numerous computer applications throughout the entire life cycle of the project. Much of the project information created during design and construction is later useful for facilities management (FM) activities during the building operation phase. When using computer integrated facilities management (CIFM), useful facilities information is captured in computer formats during design and construction so that various FM applications can use, share, and exchange the project information in an interoperable manner. This paper discusses a research effort to develop information technologies to support the development of CIFM.

The approach to supporting CIFM is closely tied to the International Alliance for Interoperability (IAI) [13] and to their efforts to develop the Industry Foundation Classes (IFCs). The IAI is a global consortium of architecture, engineering, construction and facilities management (AEC/FM) companies and software developers. Its mission is to enable interoperability among the many industry processes of all...
different professional domains in AEC/FM projects. The methodology is to enable the computer applications used by all project participants to share and exchange project information. The IAI organization is made up of several international chapters embodying industry domain committees such as architecture, structural, construction project management, facilities management, and so on. The domain committees are responsible for defining the information requirements of the industry processes within their disciplines. The IAI goal is to develop the IFCs—data classes defined in a neutral computer language that represent building project information common to all the industry processes. Using the IFCs, project information generated from design, engineering, and construction and facilities management applications can be stored in the form of computer objects to create a project model. This has tremendous value to CIFM since the project information provided in this way forms the basis for most FM applications and, thus, for CIFM environments. Therefore, the IFCs have a major role to play in the development of CIFM and other computer integrated AEC/FM environments.

However, the IFC models by themselves are not sufficient to provide the data structures needed for CIFM. The breadth of the IAI’s scope (the entire AEC/FM industry) makes the IFC models much bigger and more complex than what is needed for FM applications alone. On the other hand, the IAI’s primary goal of supporting interoperability does not address some of the details and technologies required to fully support CIFM environments. Other technologies in addition to the IFCs need to be developed to achieve the goal of CIFM. To help address this challenge, the authors are developing Facilities Management Classes (FMCs). FMCs refer to a collection of object class definitions for representing the information used in carrying out FM activities, and for supporting the sharing and exchange of FM information among FM applications within an integrated environment. The FMCs are tightly linked to the IFCs; they reference IFC objects directly where applicable, and extend IFC objects in areas where the IFCs themselves are insufficient to support FM activities. FMCs are intended to be used in conjunction with IFCs to support information exchange among FM applications and between FM and other AEC systems.

One of the research deliverables is a set of development guidelines for FMCs. This paper does not present the FMCs themselves, but rather presents a framework for the development of FMC and of CIFM environments based on IFCs and FMCs [37]. The paper starts by outlining the research objectives and deliverables. It depicts the flow of project information throughout the AEC/FM project life cycle, then presents a hierarchy of general FM functions to define the scope of FM and provide a guideline to help in developing FM information requirements. To emphasize the necessity and benefits of CIFM, the paper summarizes the types and limitations of software used in FM. The system architecture of IFC and FMC-based CIFM is demonstrated, including the system components, their roles and functionality. The development framework introduces the model requirements, types, architectures, etc., for FMCs. The paper explains the different views that enhance the model representativeness (i.e., their ability to accurately and clearly represent the required semantics of their topic area) and that support different types of software implementations. The development methodology of the FMCs is then discussed using process analysis examples. Next, the focus moves to the relationship between the IFCs and FMCs, providing a brief introduction to the IFC models. Based on the analysis of the different roles and purposes of the IFCs and FMCs, the paper explains why FMCs are needed for supporting the development of integrated FM applications. In order to obtain project data for FMC-based applications from IFC objects, IFC-to-FMC mapping mechanisms must be developed, and different IFC-FMC mapping types and methods are discussed. Finally, different FMC implementation scenarios are briefly discussed, and the paper concludes by emphasizing the roles and benefits of FMCs and the IFCs to AEC/FM projects.

The research project reported in this paper is ongoing within the Construction and Project Management Group, Department of Civil Engineering of the University of British Columbia. The research project is also associated with development work in the IAI North American Chapter’s Facilities Management Domain Committee (chaired by the first author) and Project Management Domain Committee (chaired by the third author and co-chaired by the second author). The intention of this joint research
and development activity is to avoid, where appropriate, duplicate efforts and to take advantage of both development methodologies and technologies.

2. Current practice and new directions

Facility owners and managers require large amounts of data of many different types in their operations. Typically, many different computer systems keep track of this information, such as utility and energy management, maintenance management, space management, tenant management, environmental compliance, and so on. Future FM software must be more integrated so that facilities can be managed in a more comprehensive manner throughout the life cycle of the facility. Standardization at many levels is required to integrate data and achieve interoperability among software to support FM in a practical way.

The need for standards has indeed been recognized and there are many such efforts ongoing [38]. However, most of these efforts are either academic research activities, relatively small efforts, or of regional interest. Very few have reached the stage of commercial implementations. Before any of these standards can have a significant impact on how FM is performed, they must be reasonably broadly accepted and supported, especially by software vendors. In order to provide the integration required, the standards must allow object-oriented modeling of the facilities and processes being managed. Among the evolving standards, only the IFCs developed by the IAI currently provide evidence of strong industry support in the general area of AEC and FM. This issue is further discussed under Sections 9.3.2 and 9.3.3.

The IAI is approximately 700 member organizations strong at the time of writing, organized in seven international chapters, with two more chapters slated to join soon. Many member organizations are international commercial software companies, and they are actively producing software that is IFC capable. A number of demonstrations of the latest software are held each year. At the next demonstration in November 1998, an expected 18 firms will be displaying software based on IFC version 1.5 that covers many aspects of facility design, construction and facilities management. More importantly, these systems from different vendors will be based on an IFC representation and will be completely interoperable. Meanwhile, IFC version 2.0 is nearing completion and the requirement definitions for IFC version 3.0 will be complete soon.

The IAI approach is unique in that it is a commercial effort. However, it is underpinned by a strong, formal relationship with ISO Standard 10303, STEP, in the form of a Memorandum of Understanding. From a STEP perspective, it is likely that the IAI will become the mechanism for drafting building construction domain standards for consideration and adoption by ISO. Many of the IAI members are also actively involved in STEP.

The Open Geodata Interoperability Specification (OGIS) is a related effort to produce an object-oriented standard in the GIS domain. It is under development by the Open GIS Consortium, a public, non-profit organization. The technical goals of OGIS are: to provide a single ‘universal’ spatio-temporal data and process model that will cover all existing and potential spatio-temporal applications; to provide a specification for each of the major database languages to implement the OGIS data model; and to provide a specification for each of the major distributed computing environments to implement the OGIS process model. This standard is important to the IAI effort because the IFCs currently have no geospatial capability.

3. Research objectives and deliverables

The ultimate goal of the research described in this paper is to develop technologies that can help achieve CIFM. A CIFM environment is one in which a collection of FM computer applications provide for, share, and exchange from a pool of common information about the facilities and their related entities. One of the basic requirements for a CIFM environment is the availability of AEC object information from the design, engineering, and construction of the facilities. With the development of the IFCs, it can be expected that project information will be available after the construction phase of a building project. The challenge of CIFM is then to develop the integrated systems and relevant technologies needed to ensure that the systems can obtain useful and neces-
sary FM data from the large information pool generated during design and construction and to enable interoperability between FM applications. The key technological requirement is a set of general models that represent the common information requirements of all FM functions and support data sharing and exchange among FM applications.

The objective of this paper is to describe a development framework for CIFM environments supported by the IFCs and FMCs, with a particular focus on development guidelines for FMCs. The deliverables of this project are as follows:

- A general FM function hierarchy that defines the functional requirements of FM computer applications and that is used for the development of the FMCs.
- A list of general FM elements that represent the scope of project information required from the building design, engineering, and construction for FM functions, and that are useful as a guideline for the information requirements of the FMCs.
- A system architecture for IFC and FMC-based CIFM that defines the components, their roles, and system functionality of the CIFM environment.
- A set of development guidelines for FMCs that lay out basic model characteristics and development requirements, including the following aspects:
  - FMC objectives and requirements,
  - Types of FMCs,
  - FMC architecture,
  - FM views,
  - FMC development methodology, and
  - Guidelines for the specification of IFC-FMC mapping mechanisms.

4. AEC project life cycle and scope

AEC/FM project information is created from the beginning of project planning and it develops continually during design, engineering, and construction phases (see Fig. 1). Much, but not all, of the project data generated from early project stages are useful for the later stages [38]. The end users of the project information are the building owners and facilities managers during the building operation phase. The goal of process interoperability in the AEC/FM industry is to enable information management facilities that can transfer, save, selectively drop, and distribute the project data to various project applications during the entire life cycle of the project. At the end of the construction phase, a set of project data useful for FM functions should be stored and transferred to the building operation phase. Although FM can be considered to cover the entire project life cycle from beginning to end, the focus of this paper relates to the phase after completion of building construction.

5. General FM functions hierarchy

A FM function hierarchy has been developed in the North American Facilities Management Domain.
Committee of the IAI as a guideline for developing IAI FM projects. This hierarchy also conveys the scope of facilities management functions adopted for this research.
In the hierarchy, FM functions and their processes are generally considered to have two fundamental aspects: first, a general management planning and control function, and second, a specific or identifiable FM function. The planning and control functions (see Fig. 2) are divided into scope, cost, time, work, and risk management functions, under which various sub-functions are defined [30]. The identifiable FM functions (see Fig. 3) are classified into three basic categories: maintenance and operation management, property management, and services [19]. Maintenance and operation management is further classified into the three interrelated function areas of monitoring and tracking, maintenance, alteration and repairing, and space management. Sub-functions of each of these areas are also defined.

6. FM elements

The FM functions are carried out with respect to specific elements within the facility. The FM elements are shown in Fig. 4 (again, developed by the IAI North American FM Domain Committee), which classifies the items into building systems and non-building systems, with human resources considered to be a parallel element. There was no intent to define these elements or to standardize the categorization scheme. Still, the committee members found that this identified all of the basic elements of interest for FM and that it provided a useful guide in focusing discussions about FM processes.

7. Current use of software in FM

Surveys have indicated a few basic types of software currently used in FM [11,12]. Fig. 5 illustrates the basic FM software types based on their form of integration (or lack there of). CAD applications such as AutoCAD, RUCAPS, or Computervision are primarily developed for architecture or engineering design and drafting. Thus they usually do not intrinsi-
Currently provide FM-specific functionality such as area measurement, though they are sometimes used for simple FM-related functions such as space allocation. A problem with these stand-alone CAD applications for FM is that they are typically used primarily as drawing tools, and are not used to store FM data within the drawing. GIS applications provide similar graphical interfaces as CAD, but extend the capability of associating non-graphic data with the graphics. However, the data types and structures are dependent upon specific GIS systems. Thus, it is difficult to make the GIS-based data available for other FM applications. Similarly, since the stand-alone GIS programs are not primarily developed for FM, they do not possess typical FM-specific functionality. Additionally, most FM applications require building project data from design and engineering, for which CAD-based applications are the primary tools. Although it is possible for some of the GIS systems to routinely import data from a CAD application, the imported data are still in generic CAD format without any project and FM semantics. Consequently, most GIS applications are not suitable for FM.

Another category of software used extensively in FM is databases. Many FM sites have implemented relational database management systems (RDBMSs) that support data querying and reporting for FM decision making. Since the information in the database does not include graphic representations, the users are often required to know and remember the name of the object (e.g., a space, building) as a criteria of the query in order to retrieve FM related information. This makes the browsing of the FM objects such as building elements difficult. Similar to CAD and GIS applications, general database systems do not provide specific FM functionality.

To resolve the limitations of CAD and database systems for FM use, many software vendors have developed integrated CAD-database programs that link CAD graphical elements to database records. These types of applications indeed integrate the graphics with the data to some degree, but they still face some fundamental barriers. Two basic types of graphic-data linking have been used. One is generic data-graphic linking without context constraints: graphics can be linked with any data records of any type of data structures. Many of the current CAD applications such as AutoCAD and Visio provide such a capability. The advantage of this type of linking is that it does not require application-specific database design. This allows full flexibility for users of the databases with normal query and reporting requirements; the users can design and change the database structures in any way at any time. The disadvantage, however, is that it is difficult to build FM functions based on data structures that are not specified and fixed at development time. Therefore, many FM software developers adopt an alternative linking method: linking with pre-defined database semantics. This type of linking requires the data structures of the database to be pre-defined at development time so that the interpretation of the data is understood by the programs that implement FM functions, using and producing both graphic and non-graphic data. While this type of linking enhances the FM functionality of the application, it restricts other major features of the software. First, the requirement of pre-defining the database semantic structures reduces the flexibility of the databases at the enterprise level for all other types of data query requirements, especially for responding to future changes. The pre-defined databases are almost always software specific, which means they cannot support other graphic nor non-graphic software packages. This type of CAD-database integrated software, therefore, still does not provide a completely satisfactory solution to the needs of FM users.

The last type of software for FM is non-graphic applications. Although graphic-based applications seem to be the focus of many software developers for FM, there are some non-graphic packages available for FM functions that require less graphical data or graphical user interfacing. Since this type of application only represent a small portion of FM function types, and since most of them are not integrated with other types of FM programs, they do not provide full FM functionality for FM needs.
In contrast with these existing types of FM software, the characteristics of CIFM systems are discussed in Section 8.

8. Information system architecture for CIFM

8.1. Transfer of AEC project information to FM

Much of the AEC project information useful for FM is created during the building design and construction phases. In order to make CIFM possible, this project information must be selectively transferred into CIFM systems for FM use. Fig. 6 shows how project information is transferred through the course of the AEC/FM project development. First, project information is represented as a set of inter-related AEC objects. These objects are instantiated from a set of computer models that all the project applications understand. Also, this set of models must be comprehensive and extensive enough to support all types of applications of all different domain sectors. The Industry Foundation Classes (IFCs) developed by the IAI constitute such a standard that supports industry wide applications [13]. Second, in order to allow all of the project applications to share and exchange project information effectively, an object management system is needed at the design and construction phases to facilitate the data transformation, update processes and application communications. Third, in order to retrieve all the useful data for the FM functions from this huge pool of project information, a set of models that explicitly represent the needs of the FM functions is also required. These models support the development of model mapping adapters that map the different sets of models for the development software modules that retrieve the data according to the mapping specifications. These models can also be instantiated into computer objects for the FM applications to use during building operation. FMCs are such models, developed to satisfy the information requirements of typical FM functions and processes [37]. Forth, an object management system is also needed to manage the FMCs-based FM objects, providing functionality such as object storage, queries, and updates. This type of system is also responsible for data mapping and communications between the objects and the FM applications that use them. Based on this framework, AEC project data can eventually be used and shared by various FM applications in a computer-integrated FM environment.

Fig. 6. AEC/FM information systems supported by IFCs and FMCMs.
8.1.1. System components

Fig. 7 conceptually illustrates a more detailed CIFM system component diagram at an enterprise level. The basic components are described as follows:

8.1.1.1. FMCs. FMCs are a collection of generic class definitions that represent information requirements common to all FM applications. FMCs are to support FM object creation, object interfaces, and the development of FM applications. FMCs are also used to develop the IFC-FMC software mapping engines. FMC schemas provide foundational structures for the development of CIFM systems and they will be explained in more detail later in this paper.

8.1.1.2. Object repository. An Object Repository contains FM object information needed for the FM applications. The schema of the objects and their relationships are defined by the FMCs. The Object Repository is likely to be implemented in some form of database system. The Object Repository is a conceptual term; that is, its implementation does not have to be a single repository, e.g., a centralized database. Rather, distributed database systems can also be used in which the data schema of the overall system is governed by FMCs. Although FMCs are developed using object-oriented techniques, their implementations in the Object Repository do not have to use OO databases. Traditional relational databases could be used by carefully mapping between the two different schema types.

8.1.1.3. IFC-FMC mapping engine. Software engines should be developed that map IFC objects to the FMC objects. Mapping specifications must be defined for the development of the engines that support runtime object mapping. More details about IFC-FMC mapping mechanisms can be found later in this paper.

8.1.1.4. FMC object interface. The objects supported by FMCs are accessed by various applications through object interfaces defined by FMCs. Through the interfaces, object properties can be changed and attribute values can be obtained by the applications. Since the interfaces are standardized by the FMCs, any applications that implement the interfaces can access object data. In other words, the data structures of the FMC-based FM objects are not restricted to any specific software application.

8.1.1.5. FM applications. The FM Applications are software modules that perform one or more FM functions. All of the applications require object information, which is obtainable through the object interfaces. Therefore, as long as the applications understand the object interfaces, they can be used in

Fig. 7. CIFM system architecture supported by FMCs.
the system. This means, at an enterprise level, the users are not restricted to the software from one vendor based on a single data source, and software applications from different vendors are all interoperable through the FMC interfaces. The inter-relationships between the modules are determined through their required input and output objects.

8.1.1.6. User interface. In a CIFM environment, user interfaces should be open from different sites to different users but be accessible to the same set of objects and functional modules. Internet or Intranet technologies provide these types of user interfaces and could be used in such a CIFM system. Except for user messaging or other administrative user interface functions, the basic elements are an Object Browser and Application Packages. An Object Browser is a browser that allows the user to look up all the objects and their information within the enterprise scope. It should also allow the objects to be edited. The Object Browser should also be able to access the IFC objects and perform mapping between the IFC and FMC objects. The Application Packages are containers of applications that can look up and show available software modules at runtime and allow the users to execute them. Since the application modules can be plugged in and out, the Application Packages are not pre-defined, but can be customized by the user at runtime.

The major benefits of such a CIFM system can be summarized as follows:

- It provides full application interoperability and application-object integration.
- It places no restrictions on application types, software types and database implementations.
- It is compliant with the IFCs so that important facility information can be obtained from design and construction.

9. Facilities management classes

9.1. FMC objectives

FMCs are a collection of object class definitions used to represent the common information requirements of FM functions. The classes represent real world objects in a computer-interpretable format that can be used by FM computer applications. In addition to the class definitions themselves, the development of FMCs addresses various issues of how the classes should be used to support system interoperability. The major objective of the FMCs is to support the development of multiple and interoperable FM applications in an integrated computer environment. First, FMCs should provide specifications of the data sharing and exchange requirements for the target applications. Second, FMCs should ensure data integrity, meaning that FMC-supported applications can effectively collect, share and exchange project data without losing important information during the course of data transformation. Third, FMCs should be compatible with the IFCs, ensuring that the applications supported by FMCs will be interoperable with IFC-based applications.

9.2. FMC requirements

In order to achieve the objectives for FMCs, several requirements have been defined for the development as follows.

(a) They must include the basic entities and relationships needed to represent all the elements shown in Fig. 4, including both building system elements and non-building system elements. They must also support the project management functionality associated with most FM activities.

(b) They must represent only those entities that are often needed by the FM functions; they should exclude objects generated and used only by non-FM domain applications such as construction activities or resources.

(c) They must be generic enough to represent information required by different FM functions supporting different formats and views, in order to support data sharing and exchange for different FM processes among various FM applications.

(d) They must provide specifications for mapping the FMCs to the IFCs. This ensures that objects generated by IFC-based applications can be transferred and used by FMC-based programs when data sharing across domains. This also ensures that FMC-based applications are IFC compliant so that they can also share and exchange facility information with other FM computer programs supported by the IFCs.
9.3. Types of FMC models

Real world problems are reflected in computer applications using different types of computer models at different levels of abstraction, e.g., at a conceptual level, a data level, etc. [4]. Each level may contain different types of models for different purposes. The classes that make up the FMCs also form models of the FM domain. This section explains the characteristics of these FMCs models and compares these with similar types of models developed by other research efforts.

9.3.1. FMC characteristics

The FMCs comprise a set of models at the conceptual level. First, FMCs map the semantic contexts and properties of real-world building project objects and their inter-relationships to a computer format. Therefore, a wall is represented in a computer as a ‘wall’ object (rather than a series of lines) that contains semantic attribute values, such as its thickness, material properties, etc. Second, FMCs comprise core models that provide basic classes that support the common information used throughout FM projects. Third, the FMCs comprise shared models that provide standard data structures to multiple computer applications. Once these data structures are instantiated (i.e., project-specific data values are stored), then different FM computer applications can access the data through standardized program interfaces.

On the other hand, FMCs do not define information at the data model level. First, they do not constitute application models that only meet the needs of specific processes, such as an area measurement process based on a particular measurement standard in a specific computer application. Second, FMCs are independent of specific software implementation schemes: unlike, for example, Autodesk’s ObjectARX models that only support applications that are developed to run under AutoCAD [2]. Another example is the Microsoft Foundation Classes (MFC) that only support the development of MS Windows-based applications. Third, FMCs are not database models designed specifically to support the development of database systems for a FM enterprise. For example, FMCs differ from models such as entity-relationship diagram (ERD) models commonly used for the design of a relational database, though they can be used as a reference for defining the data structures and types in the ERD model. Forth, FMCs are not classification models that attempt to classify project objects into different hierarchical categories for referencing purposes. Rather, the FMCs support linked objects with appropriate inheritance and association relationships. Moreover, the FMCs do not attempt to define a standard computer interface architecture such as COM [31] or CORBA [36], but do define standard interfaces for FM objects that can be implemented using these interfacing technologies. Additionally, FMCs are not computer programming language models such as a C++ class library, but they can be implemented using any modern computer languages. Finally, although FMCs are developed based on the analysis of typical industry processes, they are not process models that attempt to standardize FM processes and their semantic relationships, but rather to accommodate the information required by the most common practices in FM areas.

9.3.2. AEC conceptual models

A number of significant efforts to develop conceptual models for the AEC industry have been reported over the past decade [10], with the IAI’s IFC project probably representing the largest scale current effort [14–17]. Some of these conceptual models attempt to cover many project development phases or to span different domains such as the ATLAS LSE Project type Model [34], the Generic Reference Model [32], or the Unified Approach Model [3]. Other models, however, focus on more specific domain problems such as the Building Project Model [26], the General Construction Object Model [9], or the models developed in the Information/Integration for Construction (ICON) project [1] that address construction management, or the Integrated Data Model of the COMBINE project [7] that supports building design processes.

There are also relevant AEC models within the ISO Standard 10303-STEP effort [22] such as the Building Construction Core Models (BCCM) [23], which has been incorporated into the IFC core model, or Building Elements Using Explicit Shape Representation, STEP Part AP225 [24], which is a building product model that focuses on building components and their geometry. STEP building construc-
tion and IAI efforts are currently closely aligned, and a formal Memorandum Of Understanding between the two groups has been signed. All these models set good examples of developing conceptual models for the AEC industry. However, none of them are intended to explicitly support the development of interoperable FM applications used in a CIFM environment during the building operation phase. Hence, they are not developed based on identifiable FM functions, and thus they do not fulfil the requirements of FM applications.

9.3.3 Integrated FM models

Several attempts to develop explicit and integrated models for FM have been reported [33]. This section reviews examples of model for FM.

9.3.3.1 The RATAS maintenance model [28,29]. This model was developed within the RATAS project [4], where product models were developed to represent the requirements of facilities maintenance and operations. The model focused on facilities maintenance and operations.

9.3.3.2 Object-oriented model for a facility information system (FMIS) [5]. The FMIS project identified some of the fundamental requirements of an information system for FM. It stressed the need for comprehensive FM models, data repositories, and usage requirements in an integrated FM system that must be flexible to changes and must provide a uniform language and intuitive user interfaces. The FMIS model identified some of FM entities such as space, furniture, apparatus, etc.

9.3.3.3 Information system for facility management (ISFM) [27]. The models for this project were conceptual models that included both products and FM processes at a high level. This project set an example of applying identifiable facilities management processes into product model requirements. The ISFM project also attempted to formalize the FM data transformation methodology through the models for a documentation system.

9.3.3.4 Integrated facilities management information system based on STEP [6]. This project integrated a CAD system with an asset and maintenance management system and a building energy management system. The project suggested a STEP-conforming system architecture for integrated FM systems and specified a generic product data model to support the data shared by the three integrated systems.

9.3.3.5 The KBS model [33]. The KBS model was aimed at supporting the integration of FM functions [33]. The model was developed based on a set of standard national building product classification tables. The scope of the KBS model covers building products in different model views such as spatial systems, building technical systems, construction sections and construction parts. Several implementation prototypes based on the KBS model demonstrated that the model was able to support FM functions such as operation and maintenance management, tenancy agreement management, and indoor-climate calculation processes.

All these efforts demonstrate the need for explicit and integrated FM models for FM functional integration and set good model examples for future development in the area. Some of these projects also suggest and implement system architectures for an integrated FM information management system. However, the models do not attain the goals required for CIFM. The overriding reason is that none of the models are linked to standard data models such as the IFCs. This essentially precludes their ability to interoperate with applications supported by other standardized models such as the IFCs. Mapping technologies are available [25], but to date, none of them are mature enough to efficiently and practically support the development of mapping tools for two completely different model schemas. Additionally, none of these models have found general acceptance among commercial software developers. Other shortcomings of the above models for use in supporting CIFM include the following facts.

(a) Most of them support only restricted portions of the FM functions shown in Fig. 3.

(b) Most of them are not based on the information analysis of the common requirements of identifiable and typical FM processes (that is, data types are not derived from detailed analysis of FM functional process steps).

(c) Most of them emphasize the product data requirements for FM functions but fall short on
including sufficient project management concepts such as project planning, task scheduling, resource and cost management.

(d) None of the models provide software interfaces or object methods that improve the ease and efficiency of implementing the models in software applications.

9.4. FMC architecture and scope

9.4.1. FMC elements

The following is the list of the basic elements that make up the FMCs. Fig. 8 shows the schematic relationships of these elements using EXPRESS-G [20].

- Entity: represents an object type consisting of a group of attributes and associations with other entities.
- Attribute: represents a property of the entity; the change of the property value at runtime reflects the change of the object stage;
- Relationship: represents semantic associations between entities;
- Constraint: defines the scope and limits of attribute data and requirements of entity operations;
- Interface: exposes the attributes and provides services of entities with a list of associated software operations (i.e., methods);
- Method: defines the functionality of software operations and means of setting or retrieving attribute data for entities.

The entities in FMCs are also classified into two basic groups or schema according to their scope and use: Shared Element Schema and Management Reference Schema as explained in the following sections. The FMCs are primarily documented using EXPRESS [20] and EXPRESS-G methodology, while other types of documentation formats such as Interface Definition Language (IDL) [36] are also provided as necessary.

9.4.2. The shared element schema

The classes in the Shared Element Schema represent objects created from design and construction phases and useful for FM processes. These support project data that are generated by design or construction programs and that can be shared by FM applications. This schema contains classes for physical building elements. Almost all of the building elements in Fig. 4—such as wall, space, column, furniture and equipment, etc.—should be represented within the FMC shared models. Although many of the industry application concepts in this schema are also captured and modeled in the IFCs, they contain more specific data types at different detail levels here that are specific to the needs of FM applications. Fig. 9 demonstrates examples of these building element models, while Fig. 10 shows some more detailed models relating to FM requirements dealing with facility space.

9.4.3. The management reference schema

Most facilities management functions relate to general project management planning and control functions. The classes in the Management Reference Schema define management concepts such as plan,
schedule, cost, person, and so on. These are not specific to the context of AEC products or processes, but they are referenced by the shared models and essential for FM functions. Fig. 11 shows an example of the management reference models.

9.5. FMC views

Different views can be used to enhance the representativeness and to ease the implementation of the models. FMCs will be represented in three model views: Conceptual View, Specification View, and Interface View (see Fig. 12), with each containing the same model schemas discussed above. The Conceptual View presents the basic concepts captured in the FMCs with entities (i.e., classes and their relationships at a high level. No details of attributes, methods, or interfaces will be given in this view. In the Specification View, detailed attributes and data constraints associated with the entities are provided. The Interface View presents the software interfaces showing the methods that expose the attributes for each entity. Various implementation models (i.e., in an Implementation View) for FMCs such as IDL representations, C++ libraries, etc., could be derived from either the Specification View or the Interface View. However, these implementation models are not mandatory in the delivery of the FMCs.

9.6. FMC development methodology

The primary development methodology of FMCs is based on the information analysis of typical FM industry functions and processes. From this analysis,
common process information requirements will be identified and modeled in FMCs. Fig. 13 illustrates the basic development procedure. The first step is to choose a typical and identifiable FM process from one of the FM function areas as in Figs. 2 and 3. The process steps involved are then defined and an information analysis is performed for each step, resulting in the information requirements for that process. The next step is the information requirement integration, which summarizes the common information requirements based on the results of the information analysis for all processes combined—only the common information requirements will be modeled in the FMC schemas. Information modeling then begins with the ‘schematization’, where the information types are organized into appropriate schema. Entities and relationships can then be decided by analyzing the information requirements. Next, detailed entity attributes and constraints are defined and basic entity structures can be modified as necessary to incorpo-
rate the additional model elements. The model interfaces and methods are determined based mainly on the data flow of the process analysis.

The process information analysis is performed using both IFC_PDEF [16] and IDEF [18] modeling methods, taking the advantages of the easy listing of information types in the former, and of the clear data flows in the latter. As an example, Fig. 14 shows two industry processes: Occupancy Planning [15] and Asset Maintenance Planning. Each process is simplified into four major steps, and each step requires and generates information that can be matched to the model entities or their attributes. As shown in the diagram, some entities are used to define objects shared by the two processes such as the space, FF and E, and occupant. However, there are also other entities that are used as references to create management objects used by one process such as the cost, time, plan; they do not need to be shared by the other process.

The FMCs will be validated by developing the various Implementation Views, while full model validation can only be accomplished by software implementations of FM processes based on FMCs.

10. Relationship to the industry foundation classes

10.1. The IFC model architecture

The IFC model architecture [39] is based on different layers containing model schemas (see Fig. 15). The models in the Resources layer provide basic data structures to represent underlying technical or business concepts. For example, the concept of cost is defined in the IfcCost entity [14]. However, a run-time instance of the IfcCost entity is meaningless if it is not associated with some other object such as a product. IfcCost provides the data structure that can be used by any of the semantic models that require a cost attribute. The Kernel model in the Core layer defines high-level models that provide generic structures to represent the fundamental data types or relationships of the models, such as object relationships, attributes, groups, etc. The Core Extension is the first semantic layer for AEC/FM projects in the IFCs and it defines basic concepts such as products, processes, construction resources, etc. Core extension models are refinements of the Kernel. The Interoperability layer defines models that capture the semantics of a building project for information that can be shared by several different domain applications. Thus, this layer establishes the interoperability across domains, and IFC-based software applications will use these models extensively to provide data sharing interfaces. The Domain Specifications layer defines models for specific domains. Each domain is modeled in a domain schema such as Architecture, HVAC, Structural Engineering, Project Management, and Facilities Management. Adapters are also developed to support the adoption of external models (developed by other industry efforts for a specific domain) into the IFCs [39].

10.2. FMCs versus the IFCs

The IFCs and FMCs have similar goals in terms of allowing multiple applications to share and ex-

---

Fig. 15. IFC model layers.
change project information. They are also similar in modeling style and model types. However, because of their distinct target scopes, the focus and detail of the models are different. Again, the IFCs are intended to represent all the common information requirements of all industry processes across all the AEC/FM domains. Being aimed at such a broad scope, the IFCs (currently version 1.5), have not yet provided, and will not provide in the near future (i.e., within version 2.0 this year or version 3.0 next year), enough FM application models to support CIFM. A primary role of FMCs, then, is to begin to address this area, so as to provide a prototype that may eventually be adopted and adapted as an IFC model extension for FM. Nevertheless, even once the IAI turns its attention to FM with enough focus and effort to address the same FM scope as FMCs, important differences between the models may remain. For example, the IFCs will still retain extensive detail as a result of supporting so many other industry domain processes, adding significant unnecessary overhead for FM. Also, the IFCs may adopt other ‘non-FM’ perspectives on certain types of information. Because of the focuses on inter-process interoperability, the IFCs may not address as much specific detail of various FM functionality as is needed to adequately support CIFM. Thus, FMCs may remain as a distinct, but highly inter-related resource even after the IFCs address much of the same scope.

From another perspective, the difference in the impacts of the two models on the implementations of FM applications is also significant. First, since the IFCs are not developed exclusively for FM, some of the important FM specific information may not be modeled in the IFCs directly and explicitly. As a result, the implementation of a FM program directly from the IFCs would be difficult. For instance, the FM information requirements for a wall are its conceptual properties such as its materials, wall type, wall face type, and simplified geometry data such as its thickness and so on, as well as its relationship to space. However, the geometrical relationship between two connecting objects is currently modeled in the IFCs using IfcPointConnectionGeometry directly or indirectly, which is an implicit geometry data without any context meaning. Although any IFC object has the capability to represent any type of object relationship at runtime, it has no explicit conceptual meaning. In particular, an instance of IfcWall can be associated with an instance of IfcSpace using a general IFC relationship mechanism, e.g., entity IfcRelConnectsElements, but the program that queries the wall instance has no indication of the
relationship’s semantics (i.e., whether the space is being ‘bounded’ by the wall or the space ‘contains’ the wall. Although technically it is possible that the semantic relationship between the space and wall can be detected using IfcPointConnectionGeometry, practically it is not an easy task to implement this algorithm in a software program. FMCs can represent this relationship directly and be responsible for retrieving this information from the IFC objects through the mechanism that maps the IFCs and FMCs.

Second, the IFC objects contain much detail that is not needed by FM applications. For example, an instance of IfcWall entity could have associated data such as structural details, construction methods, formwork for concrete, construction resources and activities, detailed geometry data, etc. However, these types of data are not required by most FM applications. If all these data objects are carried into the FM information pool, it will not only cause unnecessary costs and difficulties for implementing FM applications, but will also degrade the runtime performance of data sharing and exchange. By using FMCs instead of the IFCs to support FM applications, the integrated FM environment can operate more efficiently.

Third, the IFCs represent all AEC/FM information combined together in one set of models of a much larger scope of what FM applications actually need. Although the IFCs cover the information requirements of different domains including FM, there is no clear view of models that explicitly and completely represents the requirements for FM applications in particular. All the FM related model elements, from entity to attribute level, are spread out throughout the entire scope of the IFCs in many of the schemas across all different layers (see Fig. 15), except the FM Schema in the Domain Schemas layer which contains only a limited set of FM related models. Consequently, it will be difficult for software vendors to develop IFC-based interoperable FM programs. Nevertheless, it is critical that FM programs be highly compatible with the IFCs if they are to interoperate with the wide range of IFC compatible software that is expected in the future.

10.3. Mapping FMCs to the IFCs

In order to achieve the goal of integrated FM application environments, project information represented by the IFC objects generated during design and construction phases should be transferred into FM phase with the FMC objects. Since the two models use different modeling architectures such as different model layers and schemas for different modeling requirements and purposes, the same or similar context meanings of certain objects might be modeled in different ways. Also, there are data types that are modeled in the IFCs in ways that most FM applications cannot use directly. These types of data requirements should thus be explicitly modeled in FMCs. Hence, mapping mechanisms between the IFCs and FMCs must be developed, and mapping specifications must be provided along with the development of the FMCs. One of the objectives of FMCs is to ensure that the FMC-supported applica-
tions are IFC compliant. This means that the FMC objects should be able to read and write IFC object information appropriately. Thus the mapping specifications should also provide guidelines for developing this functionality in the FMCs.

10.3.1. Development of mapping specifications
In order to support the development of IFC-FMC mapping mechanisms, including choosing appropriate mapping tools and designing software conversion engines, mapping specifications must be provided that outline the mapping types, the mapping model elements, and the filter specifications and functional requirements. It should be noted that the IFC-FMC mapping is model-dependent; that is, the mapping mechanisms are based on certain versions of both models. Therefore, the version number of each model must be clearly indicated in the specifications.

10.3.2. Adopting IFC Models in FMCs
While the mapping between the IFCs and FMCs may not be avoidable, the effort of developing the mapping mechanisms and runtime software engines for data mapping is a significant task. Therefore, the IFC models should be used (i.e., referenced) directly in the FMC schemas wherever appropriate and possible. Not only will this save a great deal of time for developing FMCs, but also it will ease the mapping requirements. The types of IFC models that are likely to be adopted into the FMCs are basic resource models, for example: fundamental data types like string, integer, set, or list; measurement models; common models throughout the scope of the AEC/FM industry such as organization, person, cost, time, etc.

10.3.3. Mapping Types
Although many of the elements of the two models can be mapped at modeling time (i.e., Model Mapping) based on their defined semantic meanings, Data Mapping is still necessary—that is, the runtime mapping of object data that are instances of entities in both models. The major reason for Data Mapping is that, since IFC development is an evolving effort, it will not have all the necessary entities explicitly defined in the schemas at any given point in time. For instance, the concept of an elevator is not explicitly captured in the IFC version 1.5 [14]. However, an IFC-based program could still create an elevator object at runtime by instantiating it from the IFC IfcProduct (or one of its no-abstract subtypes, e.g., IfcEquipment) (see Fig. 16). This instance can also be attached to the elevator-related attribute values defined using another IFC entity, IfcPropertySet. In this case, a FMC-based application will not be able to automatically retrieve the necessary and appropriate data from the IfcProduct instance in order to create a new elevator instance based on the FMC entity FMC_Elevator. This is because the FMC-based program does not understand about the object type of the IfcProduct-based elevator instance. Thus, a runtime mapping facility must be provided to allow the user to map the appropriate data from the IfcProduct instance to the FMC_Elevator instance. Fig. 16 depicts a Data Mapping scenario for mapping.
ping an elevator instance based on the IFC \texttt{IfcProduct} to a \texttt{FMC\_Elevator} instance based on the FMCs.

10.3.4. Mapping methods

Different mapping methods are required according to the architecture and definitions of the IFCs and FMCs.

10.3.4.1. Direct element mapping. The \textit{Direct Element Mapping} is a direct link between an IFC entity or attribute and a FMC entity or attribute (see Fig. 17). Since the IFC models should be adopted directly into the FMCs wherever possible, most of the IFC-FMC \textit{Direct Element Mappings} will be at the attribute level rather than the entity level. This is the simplest mapping mechanism with the condition that the semantic contexts of the model elements to be mapped are defined and equivalent in the two models. Although this type of mapping does not require operational data conversion as in the \textit{Functional Mapping} (see sections below), attribute data type conversions are sometimes needed. Fig. 18 shows an example of the \textit{Direct Element Mapping}. In this case, the \texttt{GenericType} and \texttt{AssignTo} attributes of \texttt{IfcFurniture} entity [14] are mapped directly to the \texttt{FurnitureType} and \texttt{User} attributes of corresponding \texttt{FMC\_Furniture} entity, and the data types of the mapped attributes need to be converted.

10.3.4.2. Filter mapping. In \textit{Filter Mapping} (see Fig. 19), an entity in the IFCs matches the contextual meaning of an entity in FMCs. However, only some of the attributes of the IFC entity match some or all of those in the FMC entity. A filter between the two entities must be defined containing a list of attributes that do not match with others. Like the \textit{Direct Element Mapping}, the \textit{Filter Mapping} does not need any functional conversions. Thus, it can be considered to be a more specific mapping mechanism than the \textit{Direct Element Mapping}. Fig. 20 illustrates an example of the \textit{Filter Mapping} between \texttt{IfcWall} and \texttt{FMC\_Wall}, where the \texttt{LayerInformation} attribute of \texttt{IfcWall} [16], which defines detailed layers data such as their offset and dimensions, is not needed by \texttt{FMC\_Wall}. Thus, it should be filtered out during this entity mapping.

10.3.4.3. Functional mapping. Unlike the previous two mapping mechanisms, \textit{Functional Mapping} deals with the problem where the attributes of the two entities being mapped do not match directly (see Fig. 21). However, the data needed by the FMC entity

\begin{verbatim}
Get IfcSpace.BoundedBy (List [0:?] of IfcSpaceBoundary);
For Each List IfcSpaceBoundary
  Get IfcSpaceBoundary.CreatedBy (IfcRelSeparatesSpaces);
  Get IfcRelSeparatesSpaces.RelatingObject (IfcBuildingElement);
  If True (Cast IfcBuildingElement to FMC\_Wall);
    Assign IfcBuildingElement to FMC\_Space.BoundingWalls (index);
End For
\end{verbatim}

Fig. 23. Function mapping procedure example.
attributes can be derived indirectly from the attributes available in the IFC entity through certain data conversion and calculation algorithms. Specifications of these algorithms should be provided in order to develop software engines for the Functional Mapping at runtime.

One example of Functional Mapping is the mapping of the geometry information of walls. Although either explicit shapes or implicit geometry can be used in IfcWall [14] to describe its geometrical properties, the concepts of wall faces are not explicitly captured in IfcWall. However, in FMC_Wall, the two wall faces are represented as two attributes, since many FM functions need to retrieve information about wall faces directly from a wall instance. The mapping from the detailed geometry data of IfcWall to wall faces of FMC_Wall requires functions to convert geometry data into wall face types. Another example is depicted in Fig. 22. Assume that FMC_Space has an attribute named BoundingWalls, which is a reference to List [0:?] of FMC_Wall. However, in IfcSpace, a wall that bounds the space is represented through a series of attribute references by IfcBuildingElement. To map the instance referenced by IfcBuildingElement to FMC_Wall, a function must be written to retrieve the entire nested referenced instance and convert the data types as necessary. Fig. 23 lists the procedure of this functional mapping.

10.3.5. Mapping tools

It is expected that good mapping tools will be helpful for developing the mapping mechanisms and software data mapping engines. Detailed mapping methods and general mapping principles have been researched and reported by a number of researchers [35]. Various model mapping tools are also available with each having advantages and disadvantages, depending on the types of the models and mapping types required [25]. The most appropriate tools can be chosen for the IFC-FMC mapping once the mapping requirements are defined.

10.4. FMC implementations

Fig. 7 demonstrates an ideal FMC implementation scenario where all FM applications, Object Repository and the IFC-FMC mapping engines implement the FMCs. In this case, all of the applications operate on, share, and exchange FM data through a single information source. The major advantage of this implementation scenario is that FM data are always up to date, and reflect the current project status. FMCs can be implemented at a lower-level as well. In particular, different FM applications specializing in different FM functions can implement FMCs so that they can communicate with each other through the FM interfaces (see Fig. 24). In this approach, no shared data repository is required, and data exchange can be done through static files such as the Clear Text Encoding of the Exchange Structure format defined in ISO 10303 Part 21 [21]. The advantage of this scenario is that it is technically easier and thus faster to implement, while the disadvantage is that objects are not updated on all the applications. For instance, a copy of a space object held by an area measurement process may not reflect the same version of information as the one being used by occupancy planning. Since different copies of one object are used by different applications at the same time, it is difficult (even impossible) to keep all copies consistent with the latest information.

11. Conclusions

The paper has presented research into the development of a framework for Computer-Integrated Facilities Management (CIFM) supported by Facilities Management Classes (FMCs) and the Industry Foundation Classes (IFCs). Since much of the information generated during the design, engineering and con-
Construction phases of a project is useful for FM functions, the project data should be selectively collected and transferred into the building operation phase for FM applications to use. The purpose of the IFC models developed by the IAI is to support data sharing and exchange among all different project applications of all industry domains. However, the paper has argued that, due to the large scope and extensive details of the IFC models that are not all needed by FM applications, the IFCs are not suitable for exclusively supporting the development of FM applications used in an CIFM environment. In response to this, FMCs that support different FM applications in an integrated FM computer environment have been proposed. FMCs, however, are not developed independently of other standard models. Rather, the development for FMCs corresponds closely with the IFC development effort with which the authors are actively involved as well. In a computer integrated AEC/FM project life cycle, the IFCs’ responsibility is to instantiate interoperable project objects during project design and construction phases, while FMCs will support data transformation from the IFC objects to the FMC instances that can be used by various FM applications during the building operation phase. The design of the FMCs is intended to ease the software implementation of interoperable FM applications desirable for CIFM.

It is anticipated that, given the availability of both the IFCs and FMCs, object management information systems can be developed that facilitate the computer integrated design, construction and facilities management during the entire life cycle of a building project. The realization of interoperable software applications based on these comprehensive industry object computer models, and their implementation in integrated computer application environments, will introduce revolutionary work practices and significant cost savings within AEC/FM industries.

Acknowledgements

Support and collaboration on portions of the work described in this paper from members of the International Alliance for Interoperability (the North American Facilities Management Domain Group in particular), Naoki Systems, and the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References