**CIFE** CENTER FOR INTEGRATED FACILITY ENGINEERING

# Engineering Test Cases to Motivate the Formalization of an AEC Project Model as a Directed Acyclic Graph of Views and Dependencies


By

**John Haymaker, Martin Fischer,**
**John Kunz, and Benjamin Suter**



**CIFE Working Paper #WP080**
**November 2003**


**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*

# ENGINEERING TEST CASES TO MOTIVATE THE FORMALIZATION OF AN AEC PROJECT MODEL AS A DIRECTED ACYCLIC GRAPH OF VIEWS AND DEPENDENCIES

JOHN HAYMAKER, MARTIN FISCHER, JOHN KUNZ, BEN SUTER
*Stanford University, Department of Civil and Environmental Engineering*
*Center for Integrated Facility Engineering, Building 550, Room 553H,*
*Stanford, CA 94305*
haymaker@stanford.edu, fischer@stanford.edu, kunz@stanford.edu, bsuter@stanford.edu

## Abstract

This paper presents industry test cases that illustrate the multi-disciplinary, constructive, iterative, and unique character of AEC projects. These test cases show that, to perform their tasks on these projects, AEC engineers construct task-specific engineering views from information in other engineering views. These engineers have difficulty constructing and integrating task-specific views on these projects today. Based on these observations, this paper proposes that engineers could benefit from an approach that provides them with simple, formal methods to iteratively construct a task-specific view from other views as needed, and control the integration of these views as the project progresses. In this way, an integrated project model emerges as a directed acyclic graph of task-specific views and dependencies. This paper defines requirements for such an approach, and discusses current project modeling approaches in terms of these requirements. While points of departure, existing approaches do not explicitly enable engineers to easily construct and integrate task-specific views and control a project model in a way that maps closely to the way AEC projects work today. To address these requirements, this paper introduces the conceptually simple Perspective Approach that enables engineers from multiple disciplines to formalize the dependency of a view on other views, and to control an evolving project model of these views and their dependencies.

## 1 Introduction

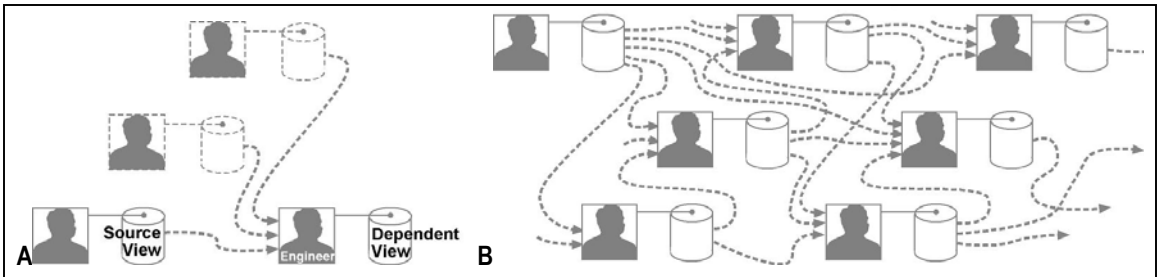
Engineers need integrated task-specific project views to help them perform their design, planning, fabrication and assembly tasks. They could benefit from project model approaches that provide them with integrated task-specific project views more quickly and accurately than current practice and theory allows. Traditionally, engineers have constructed and integrated views manually. For example, an engineer often overlays transparent drawings to assist in constructing and integrating a project's geometry in two dimensions, or an engineer may refer to a contractor's schedule when making a cost estimate. Over the past thirty years, computer-aided three-dimensional drafting (CAD) and project-modeling approaches have been developed to significantly improve the speed and accuracy with which engineers can construct and integrate task-specific geometric views. Among other things, CAD allows engineers to overlay geometric views and manually (or semi-manually, by using "design by feature" construction tools) construct dependent geometric views in response to the information in source geometric views. However, manually constructing and integrating task-specific views, with pencil or CAD, is often difficult, error-prone, and time-consuming.

Project model approaches have extended CAD approaches to address the need for automatic, integrated views of an evolving project. As discussed in this paper, some construct and integrate views through a predefined central model; others construct and integrate a federation of predefined task-specific views. To date, these approaches have been slow to be adopted on Architecture, Engineering, and Construction (AEC) projects.

Understanding the characteristics of AEC projects is essential to any effort to model these projects. This paper uses industry test cases to observe that AEC projects today are:

- *Multi-disciplinary*: Engineers from different organizations, representing different engineering criteria, form project-specific teams to design, plan, and build one-of-a-kind projects in site-specific conditions. These engineers are under time-pressure to finish their jobs quickly; they cannot be expected to anticipate or understand all other engineers' information needs when designing, planning and executing their work. They therefore construct and use task-specific views to perform their design, planning and erection tasks.
- *Constructive*: Engineers construct their task-specific views from information in other engineers' views (Figure 1A). A *dependent* view often serves as a *source* view for other dependent views. An implicit directed acyclic graph of dependencies between task-specific views forms as the design process progresses (Figure 1B).
- *Iterative*: Engineers routinely modify source views throughout an AEC project, but without being able to integrate their work with work of the other engineers on a daily basis. Therefore, engineers responsible for dependent views must become aware of modifications to source views through coordination meetings and amended documents[1]. These engineers must manually represent any implications of the modifications by re-integrating the dependent views.
- *Unique*: No two projects are alike because they are built to address a project-specific site and program with the aforementioned multi-disciplinary organization in an industry with changing and competing building technologies. Design concepts and approaches emerge within and across projects, and therefore engineers often need to construct new kinds of dependent views from changing source views.
- Finally, AEC projects, as designed and built today, are *error-prone, time-consuming, and difficult*: Manually constructing and integrating dependent views from source views causes many problems on AEC projects today.

[1]On various projects, these design versions are referred to as addenda, supplemental instructions, etc.




***A.*** *An engineer often uses discipline-specific reasoning to transform one or more source views produced by other disciplines into a task-specific dependent view that is suited to his tasks. The lines representing the dependencies between views are dashed, because they are currently not explicitly formalized. On AEC projects today, therefore, engineers often must manually construct and integrate dependent views.* ***B.*** *Dependent views become source views for other dependent views. A directed acyclic graph of dependencies between views emerges.*

*Figure 1: The dependency between views.*

The test cases suggest that to address the difficulties engineers have on these multi-disciplinary, constructive, iterative, and unique AEC projects, engineers may benefit from simple, formal methods to:

- Construct a task-specific view from other views, and
- Control the integration of that view as other engineers iteratively modify their views.

An integrated multidisciplinary project model would emerge as a directed acyclic graph of views and dependencies between views. This is a fairly simple, but potentially powerful idea that integrates product (views), as well as organization and process (the dependency between views) as the project model evolves through time.

This paper defines requirements for a project modeling approach that provides engineers with the tools needed to construct and control a project model of views and dependencies on views. This paper discusses existing AEC project modeling approaches in terms of these requirements, and concludes that, while points of departure, existing approaches are not explicitly designed to enable engineers to represent views and their dependencies in this way.

To satisfy these requirements this paper introduces the Perspective Approach, in which engineers formalize the dependency of an engineering view, called a Perspective, on other Perspectives using a composable, reusable reasoning module called a Perspector. The Perspective Approach embodies simple Management Processes that enable engineers to control the integration of their Perspectives with respect to the Perspectives on which they depend. A Project Model emerges from the iterative application of Perspectives and Perspectors. For a detailed formalization of the Perspective Approach, and how engineers can iteratively construct and control a project model consisting of geometric Perspectives and their dependencies, see Haymaker et al (2003a). For a detailed formalization of Perspectors, and how they can be modified, composed, and subsumed by engineers to construct a task-specific Perspective from other Perspectives, see Haymaker et al (2003b).

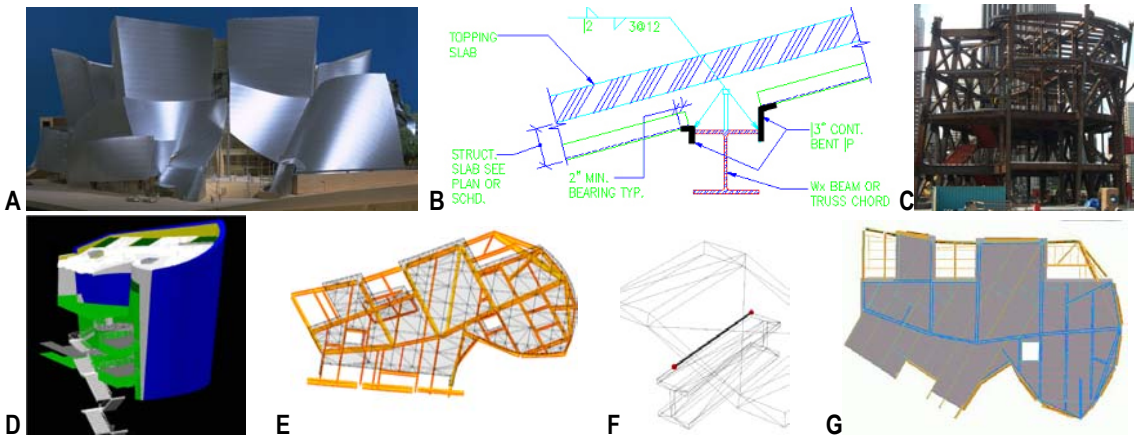
## 2 Test Cases Illustrating the Multi-disciplinary, Constructive, Iterative, and Unique Characteristics of AEC Projects

This section presents three engineering test cases to illustrate the multi-disciplinary, constructive, iterative and unique character of AEC projects today. The test cases also illustrate that to perform their design, planning and building tasks on these projects, engineers construct and integrate task-specific views from other engineers' views, and that as practiced today this is a time-consuming, error-prone, and difficult process. These test cases also suggest that if engineers could easily yet formally construct an engineering view from other engineering views, and control the integration of their view with respect to these other views, an integrated project model would emerge. This section concludes with representation, reasoning, and management requirements for testing the adequacy of alternative approaches to support engineers in constructing integrated task-specific information.

### 2.1 The Walt Disney Concert Hall: deck attachment test case

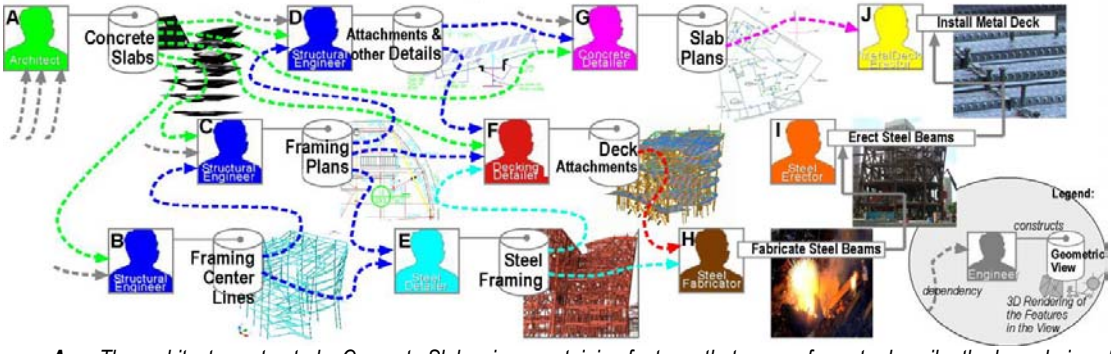
After several years on the drawing boards of architecture firm Gehry Partners, an aborted start by another general contractor, and a two-year pre-construction phase, general contracting and construction management firm Mortenson was awarded a lump-sum, at-risk contract with an aggressive required completion date enforced by liquidated damages (Post 2002). Mortenson's job was to manage the detailed design, planning, and execution of the Walt Disney Concert Hall (WDCH). As the project progressed, Mortenson subcontracted work to various engineering firms and subcontractors that specialize in specific AEC tasks. This involved an iterative design and coordination process, whereby engineers constructed task-specific project views and submitted them to Gehry Partners, Mortenson, and other engineers. From these task-specific views, other engineers constructed, and integrated task-specific views to reflect the current state of the project.

Gehry Partners' projects are known throughout the AEC industry for their advanced use of geometric models in their design and construction processes. While the form of the building is unusual (see Figure 2A), the materials and methods used to build it are conventional: structural steel, concrete, stainless steel, glass, etc. This test case involves the design, fabrication, and installation of deck attachments, which are steel angles that attach steel beams to the metal deck on which the concrete slab is poured (see Figure 2B). For clarity these illustrations present just a main stairwell and public space that the project team calls Element 2. Figure 2C shows a portion of the erected frame for Element 2. The test case describes the process that the project team used to develop and execute the design, starting from the construction of views like the visible surface model shown in Figure 2D, to views such as those describing concrete slabs and steel framing (Figure 2E), to views detailing deck attachments that connect the steel beams to the concrete slabs (Figure 2F and 2G). These views finally lead to the erection of the steel framing, deck attachments, metal deck, and concrete slab. Figure 3 describes the approximate workflow the project team executed.




***A.*** *Physical model of the WDCH (image courtesy of Gehry Partners).* ***B.*** *Deck Attachment detail view.* ***C.*** *Element 2 erected steel frame view.* ***D.*** *Element 2 exterior and interior Visible Surface view.* ***E.*** *Element 2 Roof Steel Framing and Concrete Slabs views.* ***F.*** *An engineer represents a deck attachment in a CAD layer, as a line on the edge of a beam.* ***G.*** *As-built Element 2 Roof Deck Attachments view (deck attachments are the dark lines, Concrete and Steel behind).*

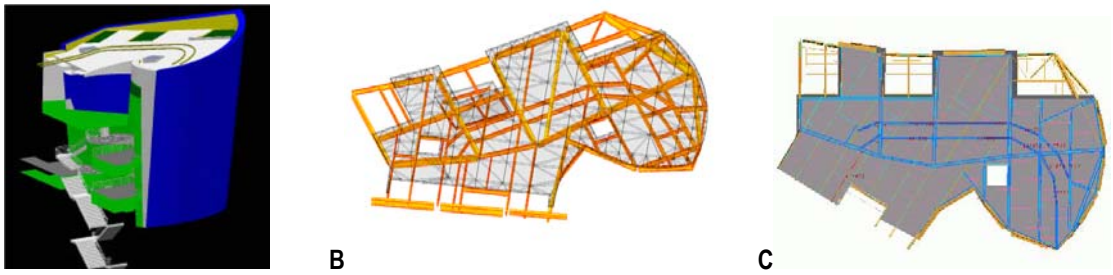
*Figure 2: Images of the WDCH for the deck attachment test case.*




- ***A.*** *The architect constructed a Concrete Slabs view, containing features that use surfaces to describe the boundaries of the project's floors. He constructed this view from information in several other views, such as the architect's visible surface view (see Figure 2D), and other views (not pictured).*
- ***B.*** *The structural engineer constructed a Framing Center Lines view containing geometric lines that describe the centerline of each column and beam. He constructed this view using information in the Concrete Slabs view and other sources of information (not pictured).*
- ***C.*** *The structural engineer constructed 2D Structural Framing views, one for each floor, from the Concrete Slabs and Framing Center Lines views. The Framing Plan identifies the size of each beam and contains some detail symbols that identify where certain construction methods are to be employed. One detail symbol refers to a Deck Attachment detail.*
- ***D.*** *The structural engineer constructed the Deck Attachment detail view containing lines and text describing how to attach a slab and beam if they are close, but the top face of the beam is not flush with the bottom face of the slab (see Figure 2B).*
- ***E.*** *The steel fabricator constructed the Steel Framing view, containing features that use surfaces to describe the boundaries of each structural steel framing member. He constructed this view from the structural engineer's views.*
- ***F.*** *The decking detailer constructed a Deck Attachments view (see Figure 2G), consisting of a geometric line along the top edge of any beam where a Deck Attachment is required (see Figure 2F). He constructed this view from the geometric Steel Framing, 2D framing plans, and the geometric Concrete Slabs. Constructing this view was error-prone, time-consuming, and difficult. It took the deck detailer over 140 hours to construct and maintain the Deck Attachments view.*
- ***G.*** *The concrete detailer constructed 2D Slab views of each slab, to prepare for installation of each slab. He constructed this view from the Concrete Slabs view and other project views not shown.*
- ***H.*** *The steel fabricator fabricated the steel beams required on the job, and welded the deck attachments onto the steel beams as required. Using information from the Steel Framing and Deck Attachments views, he determined which beam should be welded to which beam, and performed this welding in the controlled environment of the shop.*
- ***I.*** *The steel erector received the beams at the project site with the deck attachments already welded in place, and erected the steel beams using the geometric Steel Framing and other erection views (not shown).*
- ***J.*** *The metal deck erector installed metal deck on top of the steel beams and deck attachments, using the 2D Slab views.*

*Figure 3: The workflow for the design and construction team of the WDCH concrete and steel frame.*

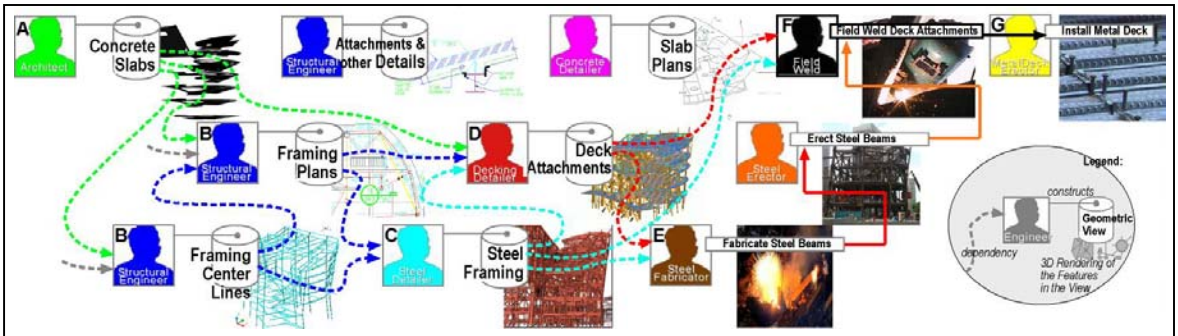
During the design development and coordination, these engineers needed to iteratively modify their views. For example, Figure 4A shows tracks for window-washing equipment that were added later in the design process. The addition of the tracks needed to propagate through the dependencies shown in Figure 3, thus requiring new and resized beams (Figure 4B) and new and revised deck attachment conditions (Figure 4C).



***A.*** *Window washing tracks are added to the roof of Element 2.* ***B.*** *The tracks (and the crane) result in additional load that requires additional steel framing.* ***C.*** *The additional steel framing requires new deck attachments.*

*Figure 4: A design modification in the deck attachment test case.*

Figure 5 illustrates the difficulties the project team encountered while constructing and integrating the Deck Attachments view. Manually finding and accurately annotating these conditions in large and complex source views, and keeping track of design changes and updating the Deck Attachments view in a timely manner was error-prone, time-consuming, and difficult. In these cases, the deck attachments could often not be welded to the beam in the fabrication shop. As a result the deck attachments required more expensive, time-consuming, and less controlled field welding.




- ***A.*** *The architect added tracks for a window-washing crane to the design (see Figure 5A).*
- ***B.*** *The addition of the window-washing crane resulted in additional loading requirements on the roof, causing the structural engineer to modify his framing views.*
- ***C.*** *The steel detailer learned of these modifications, and added beams to the geometric model where required.*
- ***D.*** *If the decking detailer did not learn of the changes or missed a specific condition, the deck attachment view did not get accurately updated.*
- ***E.*** *If the deck attachments are not accurately reflected in the Deck Attachments view, the steel fabricator did not weld the deck attachment onto the beam in the shop.*
- ***F.*** *More expensive, time-consuming, and dangerous field welding of the deck attachments was required. Over $160,000 of field welding expenses was incurred on the project due to missed conditions and difficulties controlling the propagation of design changes.*
- **G.** *Field welding often delayed the metal deck and subsequent trades.*

*Figure 5: The difficulties engineers had constructing and integrating the Deck Attachments view resulted in field welding of deck attachments. This diagram describes the propagation of the addition of window-washing tracks through the dependencies.*

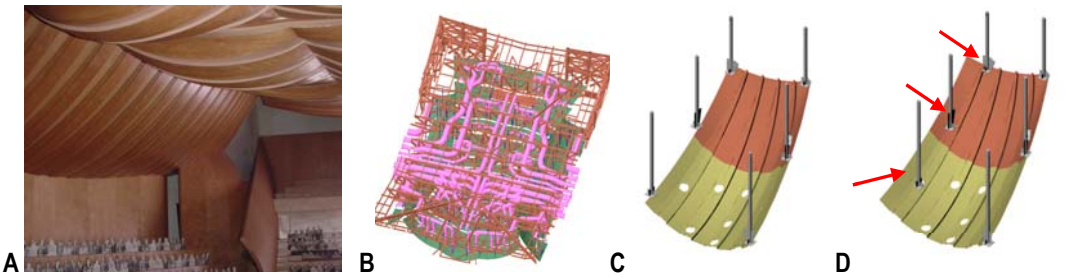
This test case provides an overview of the workflow involved in the detailed design, planning, and execution of the structural framing system of the WDCH, focusing on the deck attachments specifically. The test case illustrates that AEC processes are:

- Multi-disciplinary: The architect, structural engineer, steel detailer, deck detailer, steel fabricator, etc. specializes in different engineering tasks and also work for different organizations. They must form a project-specific team to design, plan, and build the project. Each engineer uses task-specific views to perform these tasks. The engineers are contractually responsible for the information in their view(s).
- Constructive: To perform his task, the decking detailer constructs a task-specific Deck Attachments view. He constructs this view from the Steel Framing and Concrete Slabs views. In turn, the Deck Attachments view becomes a source view for the fabricator and other engineers.
- Iterative: The architect, structural engineer, steel detailer, and deck detailer iteratively need to reconstruct their task-specific views as the design progresses. The engineers responsible for dependent views need to be notified; and they subsequently reconstruct the dependent view.
- Unique: Deck attachments are an issue on some, but not all, AEC projects. The AEC industry, software providers, or any of the engineers on the project had not previously formalized how to automatically construct a Deck Attachments view from Steel Framing and Concrete Slab views. Engineers need to construct and integrate new views not previously anticipated and formalized by computer programmers.
- Time-consuming, error-prone and difficult: constructing and integrating the Deck Attachments view took the decking detailer over 140 hours, and required over $160,000 worth of field welding.

The next two test cases illustrate the generality of these observations.

### 2.2 The Walt Disney Concert Hall: cantilevered ceiling panels test case

During the design of the daring and elegant ceiling (Figure 6A) of the main WDCH hall (Post 2003), architects, engineers, contractors, and subcontractors collaboratively designed and planned the various interrelated systems. Roof trusses, ducts, catwalks, fire sprinklers, theater lighting, and several other systems (Figure 6B) vied for a tight space above the ceiling, which is wood-faced, steel-framed, and concrete-filled for acoustic density. To aid in fabrication, the ceiling was broken into approximately 200 3m x 4m ceiling panels. Each panel, weighing in excess of 1,000 kg for acoustic density, was hung on four steel tube hangers, and each hanger was attached as close to each panel corner as possible (Figure 6C) in order to avoid *cantilever* conditions, which occur when the edge of a panel extends significantly beyond the vertical steel tube hanger designed to support it (Figure 6D). The location, number, and severity of these conditions were a factor in deciding how to frame the panels. Generally, keeping these cantilever conditions to a minimum was a design goal, but tradeoffs had to be made in order to allow all of the systems to fit into the tight space available.




***A.*** *Interior view of the main hall ceiling (taken from a physical model built by Gehry Partners).* ***B.*** *The ceiling system from above--roof trusses and ductwork are overlaid above the ceiling panels and hangers.* ***C.*** *Two ceiling panels and six hangers with no cantilever conditions.* ***D.*** *The same two ceiling panels after a design change creates four cantilever conditions (the center hanger creates a cantilever condition on each panel).*

*Figure 6: Images of the WDCH for the cantilevered ceiling panel test case.*

The engineers on this project did not construct and maintain a view of these cantilever conditions. Rather, the engineers addressed these conditions in an implicit and ad hoc fashion, based on the considerable engineering experience of the design team. This test case is therefore more speculative in nature, claiming that these engineers did not construct a view because they did not have tools to enable them to construct a new view by specifying its dependencies on other views. If such a view could be constructed quickly and accurately, it could provide useful information for multicriteria design processes. Figure 7 diagrams a simple scenario where this information could be used as a design aid for an engineering team working with three systems: ducts, hangers, and ceiling panels. As the team moves hangers to make room for certain ducts, the panels that currently have cantilever conditions are highlighted. Wishing to minimize the number of panels with cantilever conditions, the team then uses this information to decide which hangers to move when routing ducts.

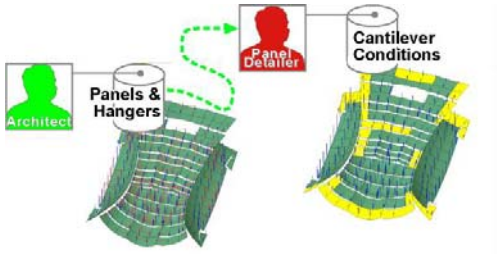


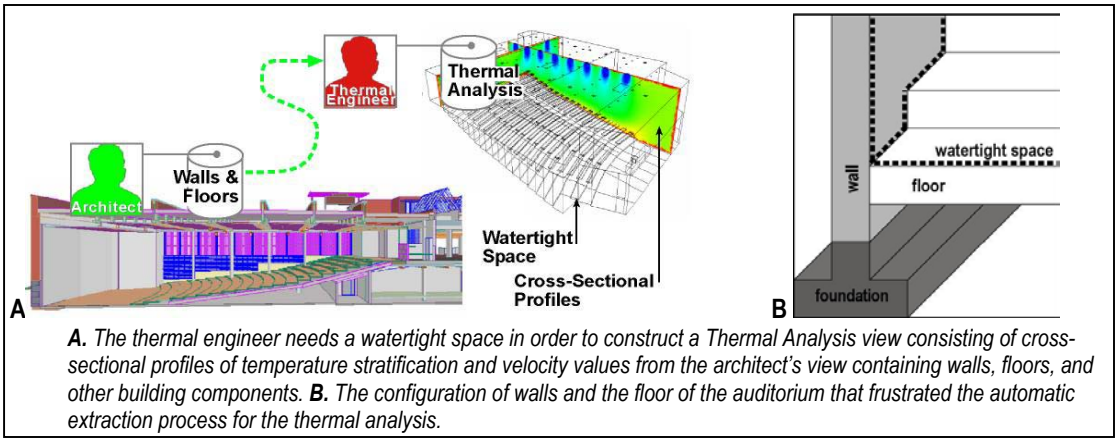

*Figure 7: A panel detailer constructs a Cantilever Conditions view from a view describing the locations of Ceiling Panels and Panel Hangers: It highlights panels that have cantilever conditions to assist in the routing of ducts.*

The cantilevered ceiling panel test case reinforces the generality of the previous observations. AEC projects are:

- Multi-disciplinary: the architect and panel detailer work for different organizations, and address task-specific engineering problems that require task-specific geometric views.
- Constructive: the panel detailer constructs (currently implicitly in his head) his Cantilever Conditions view from information in the architect's Panels and Hangers view.
- Iterative: the Panels and Hangers view is often modified, and the Cantilever Conditions view must be integrated accordingly.
- Unique: The need to construct a Cantilever Condition view from Ceiling Panels and Hangers emerges from the specific requirements of this AEC situation.
- Difficult: The engineers on this job did not even attempt to construct and integrate a formal Cantilever Conditions view. Perhaps this is because they did not have tools that enabled them to easily construct and control the integration of this view. Engineers need to construct and integrate new views not previously anticipated and formalized by computer programmers.

### 2.3 The HUT-600 Auditorium: thermal analysis test case

Frank Gehry buildings are not the only projects on which engineers construct task-specific views from other engineering views and have difficulty doing so. For example, the Helsinki University of Technology HUT-600 Auditorium is a more common structure: a US$5 million, 600-seat auditorium that was completed in February 2002. The project employed an array of design, visualization, simulation, and analysis tools as part of an international research partnership to investigate the performance of a suite of computer tools that were built to use the Industry Foundation Classes (IAI 2003). Cultural, technological, and business benefits and barriers on the project are discussed in (Kam and Fischer 2002). An example from that report discusses integration between the architect's views, consisting of floors, walls, and other building components, and a thermal engineer's view, which required a *watertight* representation of the space in order to perform a thermal analysis (Figure 8A). Because of the configuration of the walls and floors in the auditorium (Figure 8B), the thermal analysis tool's usual extraction process could not construct the required Thermal Analysis view automatically from the architect's 3D CATIA model. Extensive manual reconfiguration of the architect's views became necessary. Once such an intervention took place, the architect's views had gone through an irreversible domain-specific reconfiguration, making subsequent information sharing difficult or impossible. The thermal engineer needs to be able to describe to the computer how to construct the watertight space for this configuration of floors and walls, and control the integration of this dependent view with the architect's source views (walls and floors).




***A.*** *The thermal engineer needs a watertight space in order to construct a Thermal Analysis view consisting of cross-sectional profiles of temperature stratification and velocity values from the architect's view containing walls, floors, and other building components.* ***B.*** *The configuration of walls and the floor of the auditorium that frustrated the automatic extraction process for the thermal analysis.*

*Figure 8: Images of the thermal analysis test case.*

The thermal analysis test case further reinforces the generality of the observation that AEC projects are multi-disciplinary, constructive, iterative, and unique: Different engineering tasks require quite different views of information modeled in other engineers' views. Manually constructing and integrating these views is time-consuming, error-prone and difficult. Predetermining these views with computer programs is equally difficult.

### 2.4 Intuitions from the test cases: the opportunity to formalize the dependencies between views

Engineers use task-specific views to perform their design, planning and execution tasks. These *views* contain the information they need, structured in a way that is suited to their task. These engineers construct and integrate these views based on information in views produced by other engineers and each engineer is contractually responsible for

the information in their views. Ideally engineers could automatically construct and integrate their views. One option is to have software application programmers attempt to foresee all potential views and dependencies that engineers will want, and find ways to formalize these into central or federated project modeling systems. The engineers on the HUT-600 project tried to do this, but they were not able to formalize the necessary dependencies a priori. Foreseeing all the potential views that engineers will use has not occurred with any great rate of success to date because AEC processes are multi-disciplinary and unique: task-specific views emerge as the design is developed by multiple disciplines. We investigate whether project modeling approaches could at least be augmented by, if not founded on, the ability for engineers to construct and control their own task-specific views by specifying their own dependencies on other engineers' views, when, and where the engineering need for these views emerges.

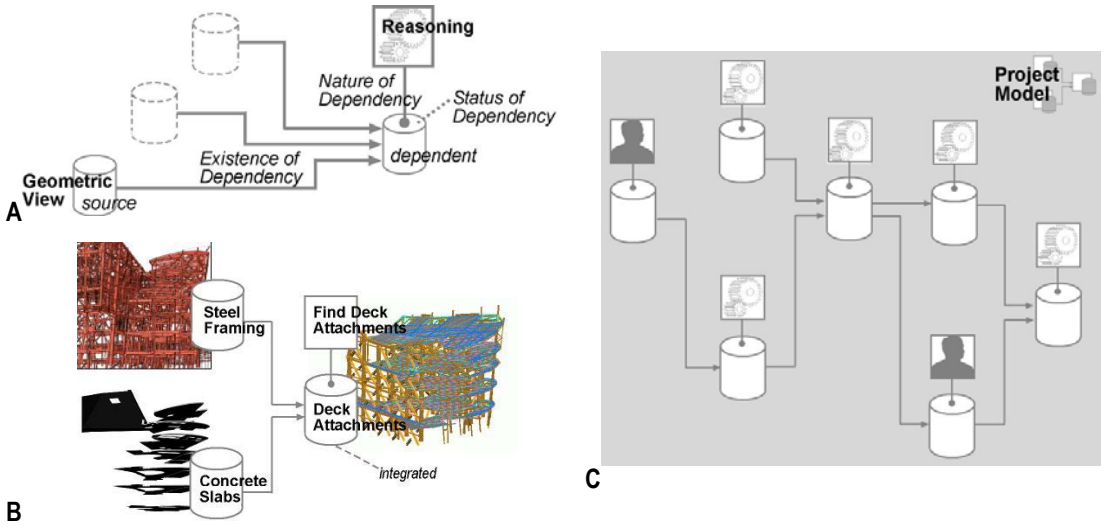
While acknowledging that the dependencies between views can often be cyclical--for example, the architect may revise the location of slabs or beams based on the number and size of deck attachments--this research investigates the conceptual simplicity of formalizing a project model as a directed acyclic graph (d.a.g.) of geometric views and their dependencies. From observations on the test case, the majority of the information dependencies are directed, and managed in the context of views; therefore this research investigates this simplicity as desirable to help manage the complexity of multi-disciplinary, constructive, iterative and unique AEC processes.

To formalize a project model as a directed acyclic graph of views and dependencies, engineers could use a simple, yet adequately expressive and formal, generic view with which they can construct their task-specific views. In the test cases each view contained a collection of geometric features. The interactions of these features in space cause engineering conditions to emerge between some but not all components. For example, only some of the slabs and some of the beams require deck attachments, and in some cases, a beam can require more than one deck attachment when it supports more than one slab. The geometric data types in these features consist of surfaces, lines, and points. Other types of views, such as schedules and cost estimates, are also constructed and integrated in the same way on AEC Projects. However this research focuses on geometric views consisting of geometric features and their spatial relationships.

Engineers could also use a method to formalize the existence, status and nature of the dependency between views. Specifically, to formalize the dependency between a view and its source views, engineers could formalize the:

- *Existence*: The source views on which a dependent view depends. For example, the Deck Attachments view depends on the Steel Framing and Concrete Slabs views.
- *Status*: Integration status of the view with respect to its source views. For example, when a slab or beam is modified, the Deck Attachments view's status should be Not_Integrated.
- *Nature*: The reasoning (automated or manual) that constructs the dependent view from source views. Engineers are generally not computer programmers; they need formal but simple methods to specify the nature of the dependencies. For example, the reasoning that the decking detailer uses to construct the Deck Attachments view from the Steel Framing and Concrete Slabs views.

Figure 9A diagrams this formalization of the dependency of a dependent view on source view(s). Figure 9B shows this formalization applied to the deck attachment test case. The reasoning, called Find Deck Attachments, analyzes beam features in the Steel Framing view and slab features in the Concrete Slabs view to construct deck attachment Features in the Deck Attachments view. Figure 9C shows that the formalization of the dependency between views can be applied iteratively to define a Project Model. The reasoning can be manual (denoted by a human icon) or automated (denoted by a gears icon).




***A**. Formalizing the existence, nature, and status of the dependency between views. **B**. Applying this formalism to the deck attachment test case. **C**. A project model can emerge from the iterative application of the formalism described in A.*

*Figure 9: Formalizing the dependency between views.*

Engineers could also use formal but simple methods to control the integration of their views, so that they can iteratively modify their views and receive notification when the views on which they depend have been reconstructed. Specifically they should be able to easily and iteratively:

1. Construct new views, information in these views, and dependencies between views (using either automated or manual reasoning).
2. Control the integration of their views with respect to the views on which they depend.

For example, the engineer responsible for the Deck Attachment view should be able to easily construct the dependencies on the Steel Framing and Concrete Slabs views, be notified when these source views are modified, and be able to (re) construct the deck Attachments view. Other engineers should be able to construct and control dependent views of the Deck Attachments view.

The following summarizes our intuitions into requirements for a project modeling approach that addresses the characteristics of AEC projects observed in the test cases and overcomes the difficulties encountered by the engineers on these test cases by enabling engineers to formalize a project model as a directed acyclic graph of views and

dependencies. These requirements are also useful to determine whether alternative project modeling approaches enable engineers to easily construct and control views and their dependencies. A project model approach should formalize:

- **Representation:**
    1. A generic engineering view containing the formalisms to represent an engineer's task-specific views. It should be simple enough to be understood, yet also contain adequate expressiveness to describe a wide range of task-specific needs; for example, a generic view that an engineer can use to construct, among other views, the Steel Framing, Concrete Slabs, Deck Attachments, Ceiling Panels, Panel Hangers, and Cantilever Conditions views.
    2. The generic existence, status, and nature of the dependency of a dependent view on its source views. For example, an engineer should be able to represent the dependency of the Deck Attachments view on the Concrete Slabs and Steel Framing views.
- **Reasoning:**
    3. Generic reasoning that constructs a dependent geometric view from source geometric views to enable engineers to integrate their task-specific views with the existing project information that is relevant to them. The approach should contain methods for engineers (as opposed to computer programmers) to define this reasoning, and thus extend the project model in ways that cannot be anticipated; for example, reasoning that engineers can use to formalize the nature of the dependency by specifying how to construct a Deck Attachments view from the Concrete Slabs and Steel Framing views.
- **Management:**
    4. Generic methods for engineers to iteratively construct (add, modify, and delete) instances of 1, 2 and 3; for example, tools to enable an engineer to construct the Deck Attachments view by specifying its dependence on the Concrete Slabs and Steel Framing views. Other engineers should then be able to construct new views that depend on the Deck Attachments view.
    5. Generic methods for engineers to control the iterative construction and integration of this graph of views and their dependencies as it is iteratively modified in a multi-disciplinary way; for example, tools to enable an engineer to construct the Deck Attachments view with respect to the Concrete Slabs and Steel Framing views and to control the integration of these views.

The next section discusses other AEC project modeling approaches with respect to these requirements.

## 3 Related Research: Project Modeling Approaches

An increasing number of researchers (Van Nederveen and Tolman 1992, Howard et al 1992, Eastman and Jeng 1999, Clayton et al, 1999, Turk 2001) and industry professionals (Zamanian and Pittman 1999, Newton 2002, Bentley 2003) are recognizing the need for model evolution to allow engineers to construct new information, and to define the existence, status and nature of the dependencies between information. Given the project model requirements described in Section 2.4, this section discusses related computer-based project modeling approaches aimed at providing integrated views of an evolving project. The approaches can be grouped into two categories:

- Representational Approaches that develop generic and specific schemas, which engineers can use to construct and relate AEC information. After an overview of approaches in this category, the Industry Foundation Classes (IFCs) are discussed as an example of how representational approaches relate to the requirements.
- Reasoning and Management Approaches that formalize ways to represent AEC information, but also formalize the nature of the dependency within this information. After an overview of approaches in this category, EDM-2 is discussed as it relates to the requirements.

While current representation and reasoning and management approaches are points of departure that satisfy many of the requirements, they are not designed to enable engineers to easily construct a new view from other views, or to iteratively control the integration of an evolving project model of these views and their dependencies. Conceptually, most prior approaches provide methods to construct views from a central pre-defined model. The approach based on the requirements and formalized in this research lets a project model emerge from the iterative construction and integration of task-specific views.

## 3.1 Representation approaches to integrated AEC project models

Some researchers approach integrated project modeling by formalizing pre-defined project- or industry-wide schemas (i.e., Björk 1987, Gielingh 1988, Step 2002, IAI 2002). These approaches formalize ways for engineers to construct representations of typical AEC components (i.e., beam, slab), attributes (i.e., beam type, geometric description), and relationships (i.e., connected-to, contained-in-structure). Engineers construct a central model using this schema, and using knowledge of this schema construct task-specific integrated views of this model. Other representational approaches investigate generic schemas containing generic concepts (Howard et al 1992, Stouffs and Krishnamurti 1997, Hakim and Garrett 1997, Clayton et al 1999, Van Leeuwen 2002) that engineers use to construct task-specific information and views. Representational approaches satisfy the representation requirements for a generic view, and can be used to formalize the existence and status of the dependency between views, however they do not formalize the nature of the dependency between views, and they do not address the reasoning and management requirements. The Industry Foundation Classes (IFCs) are an emerging industry standard for a project model schema that builds on many of the earlier representational approaches. The next section reviews the IFCs to illustrate the extent to which representational schemas satisfy the requirements.

### 3.1.1 The Industry Foundation Classes

The IFCs are developed by the International Alliance for Interoperability, according to the IFCs 2X technical guide:

> *"To support information about AEC/FM generally so as to enable the sharing of information between disciplines. It is a key objective of the model that it is interdisciplinary and consequently the focus of the model is on this level of information exchange and sharing. It is not intended to support a particular discipline or the application requirements of such a view. Neither is it intended to provide the basis for a database that supports the application requirements of a particular discipline, although it could fulfill such a purpose in certain circumstances." (IAI 2002)*

Table 1 summarizes how the IFCs relate to the requirements. Figure 10 shows a part of the schema defined in IFCs 2X that an engineer could use to relate an *Ifcbeam* feature to an *Ifcslab* feature through a *Ifcrelconnectselements* relationship. Using an *Ifcconnectionconstraint*, an engineer can construct a dependency on this relationship and assign the type of joint to the constraint (the engineer would need to extend the *ifcjointenumeration* to include a *deck attachment* type). The engineer can construct the beam's physical boundaries using *ifcsurfaces* and can define the surfaces of the beam and the slab that are to be connected by the deck attachment. Engineers can construct views of these slabs and beams through predefined views such as *ifcbuildingstorey* (not shown), and *ifcbuilding*.

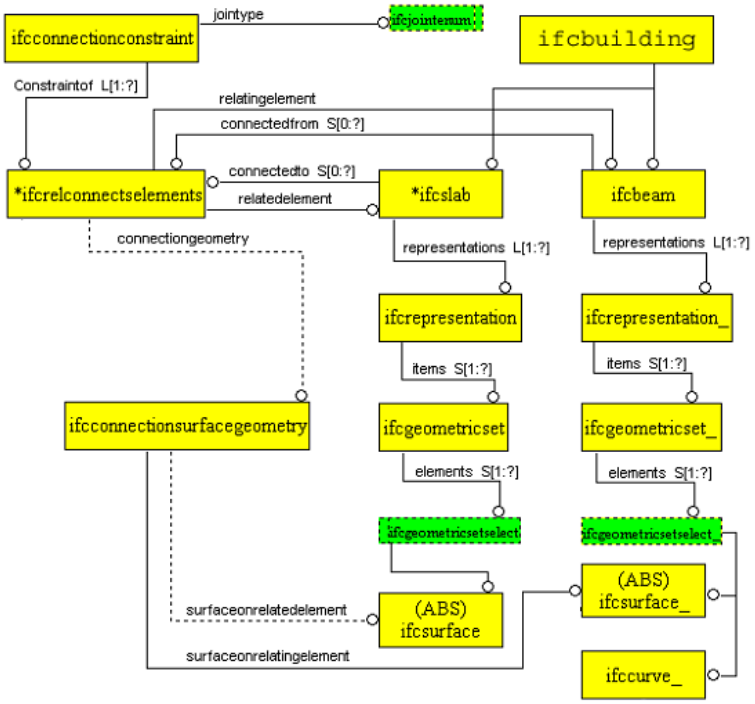



*Figure 10: Using the IFCs to represent the connection relationship between a slab and beam.*

The IFCs do not contain a concept of deck attachments or a Deck Attachments view. However, they contain generic concepts, such as *Ifcgroup*, *Ifcbuildingelement*, and *Ifcsurface* that can be used to describe geometric views, which contain geometric features, and thus satisfy requirement 1. The IFCs do not formalize a depencency relationship between views, however they contain the generic *Ifcconstraint* relationship that could be specialized for this purpose. This would enable an engineer to formalize the existence and status of the dependency of a Deck Attachment view on a Concrete Slabs and Steel Framing view. However, the IFCs do not contain a formalization of the nature of the dependency, therefore requirement 2 is only partially satisfied.

The IFCs make no commitment about reasoning or management processes, therefore the IFCs alone do not satisfy any of the subsequent requirements. The purpose of adopting a standard schema, however, is that software companies develop software that engineers can use to construct and control a model defined using this schema. Therefore this review continues by taking into account various software that can be used on an IFCs model.

Structure Query Language (SQL): Engineers can construct dependent views of an IFCs model using generic reasoning such as SQL (Date and Darwen 1993). SQL contains numerical and textual reasoning that engineers can use to construct dependent views that are not explicitly defined in the model schema. However, SQL does not explicitly formalize the existence and status of dependencies in the project model, nor does SQL manage these dependencies. That is, even if an SQL query could be formalized to construct a Deck Attachments view from Concrete Slabs or Steel Framing views (SQL currently lacks the geometric operators to do so), the need to integrate the Deck Attachments

view when the slabs and beams views are modified cannot be explicitly formalized, or managed. The deck attachment view does not become an integrated part of the model from which other views can be constructed and integrated.

Solibri Model Checker: Engineers can use Solibri Model Checker (Solibri 2003) to construct a dependent view based on various preprogrammed, but user-modified constraints (e.g., to detect whether beams and slabs overlap in space) to construct a dependent view highlighting where the violations occur. Solibri Model Checker does not yet enable an engineer to construct a dependent view not previously formalized by the system programmers, and again, the existence and status of the dependency between views is not explicitly formalized or managed.

IFCs compatible CAD tools: Tools such as Archicad (Graphisoft, 2003) for architects and MagicCAD (Progman 2003) for heating, ventilation and air conditioning (HVAC) engineers enable these engineers to manually construct task-specific IFCs views. These CAD programs provide feature construction tools that help engineers, for example, construct a beam, slab, or duct with a few mouse clicks. Again, these tools do not explicitly formalize or manage the existence and status of the dependency between views.

BSPro: BsPro (Granlund 2003) is a model server that stores the IFCs model, and provides some tools for programmers to construct views of and write information to this model. Again, the existence and status of the dependency between views is not explicitly formalized or managed.

These external applications partially satisfy requirement 3 because they enable engineers to construct many useful dependent views from information in the source view (the IFCs model). However none of these tools enables engineers to construct new views easily by constructing new dependencies on other views and control the integration of the views as they are iteratively modified; therefore requirements 4 and 5 are not satisfied.

| **Representation:** | **IFC** |
|---|---|
| 1. A generic engineering view | IfcGroup,Object,Surface |
| 2. Existence, nature, and status of the dependencies between views | IfcConstraint |
| **Reasoning:** | |
| 3. Construct a dependent view from source views | External Applications |
| **Management:** | |
| 4. Construct views and dependencies on other views | |
| 5. Control the integration of new and modified views | |

| **Legend** |
|---|
| Does not satisfy |
| Partially satsifies |
| Satisfies |

Table 1: Comparing the Industry Foundation Classes and related applications to our requirements.

Using representation approaches (with some extensions), engineers can manually construct many task-specific views, and manually specify the existence and status of the dependency of information in these views on information in the IFCs model. Current representation approaches do not explicitly formalize the nature of the dependencies between views, and, even with the help of external applications, they do not formalize reasoning and management processes to enable an engineer to easily construct and control a project model consisting of task-specific views and dependencies.

### 3.2 Reasoning and management approaches to project-modeling for AEC projects

To address the limitations of relying solely on representational schemas to support integration, other researchers investigate reasoning and management approaches. Many in this reasoning and management category, such as Cutkoski et al (1993), Eastman and Jeng (1999), Haymaker et al (2000), Sacks (2002), and Autodesk (2003) construct and control dependencies between information in a central model. Others, such as Khedro and Genesereth (1994), Rosenman and Gero (1996), Mackellar and Peckham (1998), Sriram (2002), and Bentley (2003) construct and control dependencies between a federation of task-specific views.

While some report success constructing and controlling dependencies within the context of single domains (i.e., Sacks et al 2003, Tow and Harrison 2003, Reina 2003) reasoning and management approaches are not being widely used in the AEC industry to integrate the work of multiple disciplines. As currently formalized, they do not readily support the multi-disciplinary, constructive, iterative, and unique nature of AEC projects.

As an example of the reasoning and management approaches, the next section compares EDM-2 to our requirements, illustrating that EDM-2 partially satisfies all of the requirements in that it enables computer programmers to construct and control dependencies between information in a central model and task-specific views, or directly between information in task-specific views. However, it does not provide the representation, reasoning and management concepts to provide engineers with the guidance they need to construct and control a project model consisting of views and dependencies between views, and thus meet our requirements.

#### 3.2.1 The Entity Data Model-2

EDM-2 is a product modeling and database language intended to support the "evolution of a product model based on multiple application views and mapping between them and the central building model," and includes "explicit specification and automatic management of integrity between a building model and various views" (Eastman and Jeng 1999). Figure 11 shows how Eastman and Jeng diagram this approach.

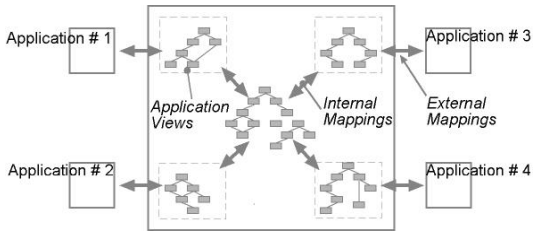



*Figure 11: EDM-2: a product model based on multiple application views and a central model consisting of Design Entities (gray rectangles). Image is redrawn from Eastman and Jeng (1999).*

The test cases illustrate that engineers could benefit from a generic engineering view in which they can formally describe task-specific views: EDM-2's primary concept is called a Design Entity (DE). A DE can contain data types and other DEs. Therefore a DE can be used to represent a view, features in these views, and information in these features or any other concepts. EDM-2 formalizes a Composition, which can also be used to contain DEs. EDM-2 provides the tools with which to construct a view, and therefore satisfy requirement 1. The test cases also illustrate the need to formalize the existence, status, and nature of the dependency between views. EDM-2 formalizes Constraints, which

are used to formalize the existence and status of the dependencies between DEs in the model. EDM-2 formalizes the nature of the dependencies using Maps (Eastman and Jeng 1999) or Operations (Eastman et al 1997) that construct DEs in the model based on other DEs in the model. EDM-2 does not explicitly guide engineers to formalize the existence, status and nature of the dependencies of a DE on other DEs, therefore EDM-2 only partially satisfies requirement 2. Figure 12 shows a diagram that describes the relationship between DEs and Operators (Maps, described in Eastman and Jeng 1999, are similar). An Operator can construct several DEs, and several Operators can construct one DE. EDM-2 does not explicitly formalize reasoning that constructs a DE from other Des. However, the existing mechanisms (DEs, Constraints, Maps / Operators) could be specialized to do so; therefore, requirement 3 is partially satisfied.

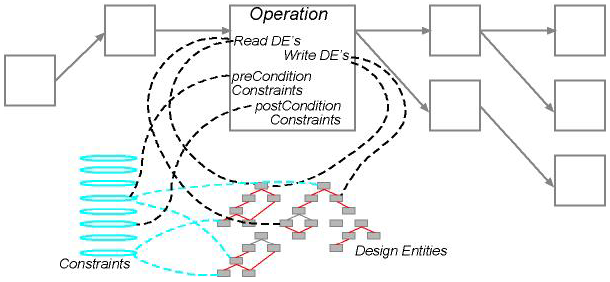



*Figure 12: EDM-2 formalizes dependencies using a graph of Operations that iteratively access and construct a central model of Design Entities. The Operators can be controlled by Constraints. (Adapted from Eastman et al (1997)).*

The test cases illustrate that engineers require methods to construct new task-specific views by formalizing the dependency on other task-specific views. Currently system programmers are required to construct the dependencies and the views in EDM-2; therefore requirement 4 is only partially satisfied. The integration system implemented in EDM-2 is designed to use a central model to integrate task-specific views: an Operation / Map constructs information (DEs) in a task-specific view -- the integration system propagates this construction to the central model -- the central model contains Constraints that monitor these DEs and use Maps to construct any dependent DEs -- these constructions are then propagated to other task-specific views using other Constraints and Maps. While EDM-2 controls a project model of views organized around a central model, EDM-2 does not currently provide engineers with the guidance needed to control a project model that consists of a graph of views and their dependencies on other views. Therefore requirement 5 is only partially satisfied.

| **Representation:** | **EDM-2** |
|---|---|
| 1. A generic engineering view | DEs, Compositions |
| 2. Existence, nature, and status of the dependencies between views | Constraints, Operations / Maps |
| **Reasoning:** | |
| 3. Construct a dependent view from source views | Operations / Maps |
| **Management:** | |
| 4. Construct views and dependencies on other views | System Programmers |
| 5. Control the integration of new and modified views | Operations |

| **Legend** |
|---|
| Does not satisfy |
| Partially satsifies |
| Satisfies |

*EDM-2 partially satisfies the requirements needed to formalize and control the dependencies between views. However, EDM-2 does not contain enough representation, reasoning and management commitment to adequately guide engineers in formalizing a project model consisting of task-specific views and their dependencies*.

Table 2: Comparing EDM-2 to our requirements.

EDM-2 is a database language that contains mapping techniques to integrate information in views that are organized around a central building model. Who constructs and maintains the central model in EDM-2, and what its contents are, remains an open issue. In addition, EDM-2 makes the assumption that all modifications to the central model are permitted; however, our test cases show that each engineer is responsible for certain project information. Resolving access rights in a central model for a multi-disciplinary project is also an open issue. While it seems possible that the mechanisms in EDM-2 could be used to construct information in views from other views, this has not been the focus of EDM-2, and has therefore not been formalized or tested. EDM-2 does not provide guidance for how engineers from multiple disciplines can construct and control a project model in this way.

### 3.2.2 Other Reasoning and Management Approaches

Parametric approaches are reasoning and management approaches in which engineers can define numeric or symbolic equations that depend on other numeric or symbolic equations (Shah and Mäntyla 1995). When solved, these equations realize feasible designs. These approaches commonly use a graph, often called a history tree, to structure the dependencies between concepts (Serrano and Gossard 1987). Shape Grammars (Stiny 1980) provide a computational approach to the generation of designs, although they have not been developed to enable engineers to construct and control the integration of task-specific views from other engineers' views. Commercially available parametric modelers for the mechanical engineering industry such as CATIA (Dassault 2003) currently provide tools to assist engineers to construct 2D sketches from which the system can parametrically construct 3D Features, and other tools to parametrically specify how to position physical components in relation to other components. For example, an engineer can parametrically relate the bottom face of a slab's geometry to the top face of a beam's geometry, such that moving the beam will also move the slab. Some systems employing these parametric techniques are being commercially introduced specifically for the AEC industry, such as Tekla's Xsteel, Autodesk's Revit, and Microstation's TriForma. Some parametric approaches in the mechanical engineering domain formalize tools that are geared towards enabling fast development time for reasoning that constructs and controls dependencies between information. For example A-Teams (Talukdar et al 1996) is a problem solving architecture in which agents are autonomous and modify each other's trial solutions. Exemplars (Bettig et al 2000) are used to describe complex situational patterns and extract information of interest. Both use reasoning to transform information from one view into information in other views. DOME (Abrahamson et al 2000) provides a publish, subscribe, and synthesis framework to develop a service marketplace for the mechanical engineering industry. This approach enables a system integrator (who does not need to be a computer programmer) to build the dependencies between information in various engineering views. These approaches match the spirit of the requirements for AEC project modeling derived from the test cases; they do not contain explicit formalisms to guide AEC engineers in constructing and controlling the integration of a view from information in other views.

### 3.3 Summary: comparison of project modeling approaches to requirements

The test cases above illustrate that, in many cases, formalizing a project model as views and dependencies is an appropriate approach for representing and integrating multidisciplinary design information. Such an approach appears to map closely to the way AEC engineers work today. Figure 13 schematically illustrates that representational approaches for AEC project modeling generally formalize more representation concepts than the requirements demand, but do not contain sufficient reasoning and management formalization. For example, the IFCs formalize many representational concepts, although they do not explicitly formalize the existence, status and nature of the dependency of a view on other views, and they formalize virtually no reasoning or management concepts. Current reasoning and management approaches for AEC project modeling provide techniques to construct and control the integration of information in a model, however they do not formalize sufficient representation, reasoning, and management to guide engineers in the construction and control of an evolving, multi-disciplinary project model consisting of views and dependencies. For example, EDM-2 formalizes very general Representation (Design Entity, Constraint), Reasoning (Operator), and Management (Integrity Management Subsystem). The requirements demand more specific representation (views, existence, status, nature of dependency between views), reasoning (construct one view from other views) and management (control views and dependencies).

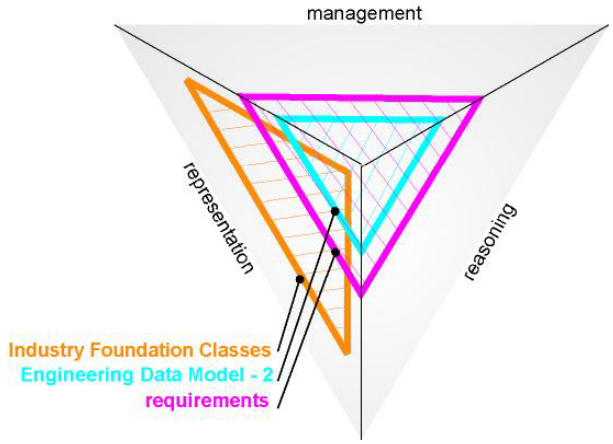


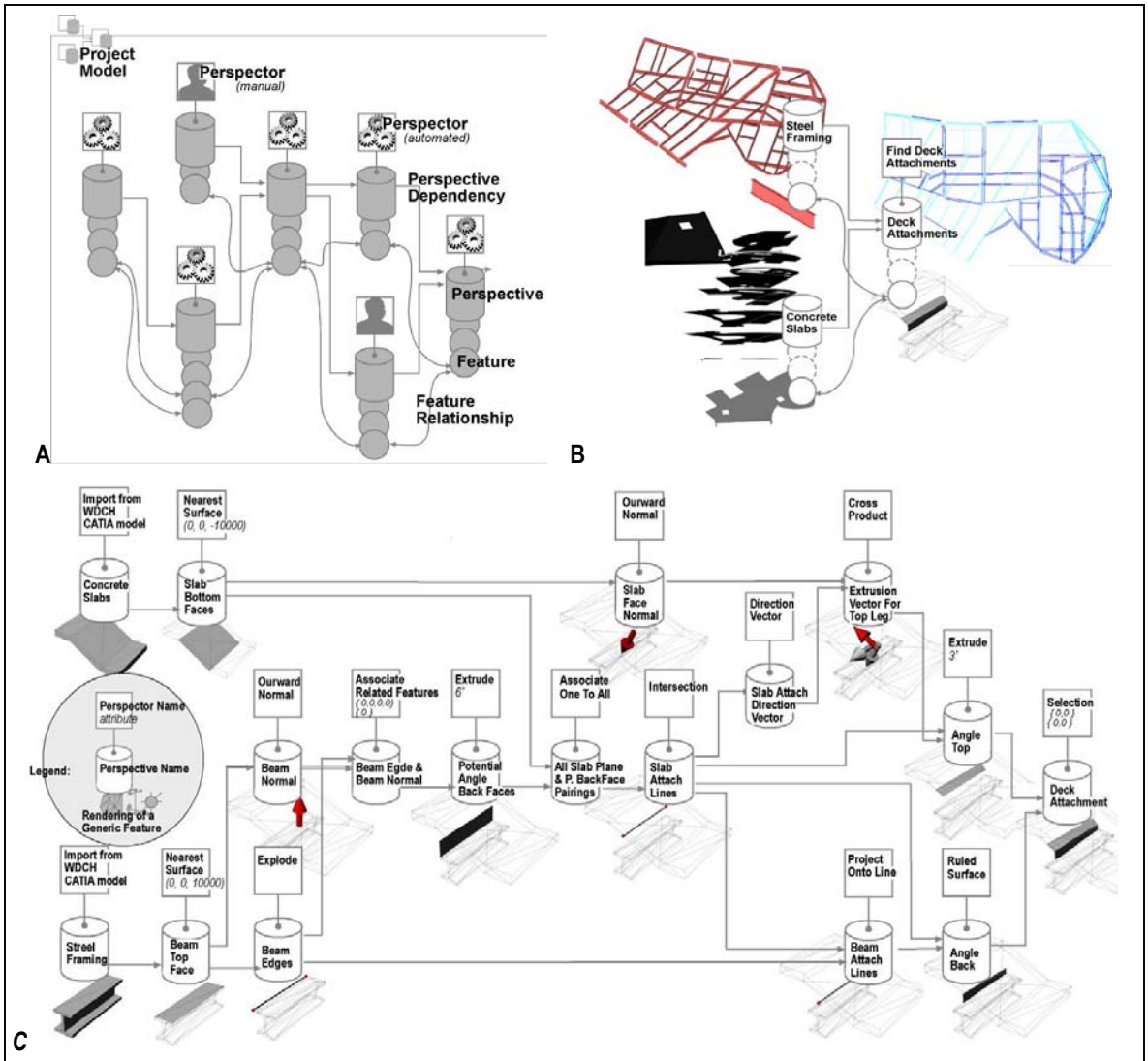

*Figure 13: Qualitatively relating EDM-2, the Industry Foundation Classes, and the requirements, in terms of amount of representational, reasoning, and management formalization specified by the requirements.*

The next section introduces the Perspective Approach, which is founded on these requirements.

## 4 The Perspective Approach

The Perspective Approach formalizes a project model as a directed acyclic graph of views and the dependencies between views. In the Perspective Approach (see Figure 14A) a task-specific view is called a Perspective. A Perspective can contain the information needed to describe the project for a specific task, satisfying requirement 1. A Perspective also contains a formalization of the existence, status, and nature of its dependency on other Perspectives, satisfying requirement 2. Engineers use composable, reusable reasoning modules, called Perspectors, to formalize the nature of the dependency of a Perspective on other Perspectives. A Perspector analyzes the information in source Perspectives to construct information in the dependent Perspective, satisfying requirement 3. A Perspector can be automated, or it can provide tools to an engineer to construct this information manually. Because a Perspector's input and output are both Perspectives the approach can be applied modularly. Engineers can use this modularity to compose Perspectors to achieve complex transformations of source Perspectives into a dependent Perspective. Perspectors can be subsumed into a higher-level Perspector that represents the entire transformation performed by lower level Perspectors. A Project Model develops over time as a directed acyclic graph of dependencies between Perspectives, where the nature of these dependencies is represented by Perspectors. The composable, modifiable, and subsumable formalization of Perspectives and Perspectors enables engineers to formalize their own dependencies of a view on other views, satisfying requirement 4. Before a Perspector constructs a dependent Perspective, it checks that the Integration Status of all source Perspectives is marked as *Integrated*. When a Perspector completes, it assures that the Integration Status of all dependent Perspectives, iteratively down the graph, is marked as *Not_Integrated*. These simple management processes enable engineers to control the integration of these Perspectives, satisfying requirement 5. By design, the Perspective Approach satisfies the representation, reasoning, and management aspects of the requirements.

For example, Figure 14B applies the Perspective Approach for the deck attachment test case described in this paper. This test case involved constructing and integrating geometric views containing Features. The figure therefore describes Perspectives containing geometric Features that contain relationships to Features in other Perspectives. An engineer (in this case the first author of the paper) uses a Find Deck Attachments Perspector to analyze geometric Features in Steel Framing and Concrete Slabs Perspectives to construct geometric Features in a Deck Attachments Perspective. The Deck Attachments Perspective can become a source perspective for other dependent Perspectives (not shown). An engineer can use the Perspective Approach and reusable geometric Perspectors to formalize the Find Deck Attachments Perspector. The Perspectors composed in Figure 14C can then be subsumed by the Find Deck Attachment Perspector.




***A.*** *A geometric Perspective is a task-specific view containing Features that use geometry (not shown) and relationships to other Features to describe task-specific engineering concepts. A Perspector is a reasoning mechanism that analyzes the Features in source Perspectives (through the source Perspective relationship) to construct Features in the dependent Perspective.* ***B.*** *Applying the Perspective Approach to the Deck Attachment test case: The Find Deck Attachments Perspector analyzes the slab Features in the Slabs Perspective and the steel framing Features in the Steel Framing Perspective to automatically construct deck attachment Features in the Deck Attachments Perspective. The Find Deck Attachments Perspector also relates slab and steel-framing Features to associated deck attachment Features.* ***C.*** *Composing a directed acyclic graph of low-level Perspectors and Perspectives to analyze slab Features in a Concrete Slabs Perspective and steel framing Features in a Steel Framing Perspective to construct deck attachment Features in a Deck Attachment Perspective. An engineer can then subsume This Perspector Graph into the Find Deck Attachments Perspector shown in B. These Perspectors are described in detail in Haymaker et al (2003b).*

*Figure 14: The Perspective Approach.*

| Representation: | Perspective Approach |
|---|---|
| 1. A generic engineering view | Perspective, Feature |
| 2. Existence, nature, and status of the dependencies between views | Perspective, Perspector, Status |
| **Reasoning:** | |
| 3. Construct a dependent view from source views | Perspector |
| **Management:** | |
| 4. Construct views and dependencies on other views | Management Processes |
| 5. Control the integration of new and modified views | Management Processes |

| Legend |
|---|
| Does not satisfy |
| Partially satsifies |
| Satisfies |

Table 3: Comparing the Perspective Approach to our requirements

Haymaker et al (2003A) detail the formalization of the Perspective Approach for the WDCH test cases, describing how to iteratively assemble geometric engineering views, called Perspectives, into a network of dependencies, and how to construct and control the integration of this evolving model as the project progresses. The paper shows how the approach can help overcome the difficulties engineers face in constructing and integrating views today. Haymaker et al (2003b) formalize the reusable, composable, subsumable reasoning modules, called Perspectors, which engineers use to automatically, or manually, construct task-specific 3D views, called Perspectives, from other engineering Perspectives. The paper shows that engineers can select from a reasonably small number of predefined, reusable geometric Perspectors and compose and customize them into a directed acyclic graph that specifies automatic or manual transformations of many source geometric Perspectives into better, faster, and cheaper dependent geometric Perspectives than current practice allows. Evidence for the power and generality of the Perspective Approach is provided by implementing a software prototype on two of the engineering test cases presented in this paper from the design and construction of the Walt Disney Concert Hall.

## 5 Discussion

The test cases presented in this paper show that engineers today have a difficult time constructing and integrating geometric views even on projects using state-of-the-art tools like CATIA. This is due partly to the multi-disciplinary, constructive, iterative, and unique qualities of AEC projects. Our intuition is that, to address these difficulties, engineers could benefit from formal, simple methods to construct and control the dependencies between views. This paper: (1) proposes to formalize a project model as a directed acyclic graph of views and their dependencies, (2) shows that representation approaches for AEC project modeling such as the Industry Foundation Classes define many useful ways to represent task-specific concepts, but do not currently formalize or manage the nature of the dependencies between geometric views in any appropriate systematic way, and (3) shows that reasoning and management approaches for AEC project modeling, such as EDM-2, formalize the dependencies between concepts, but do not formalize simple methods for engineers to explicitly construct and control the dependencies between views.

We investigate whether project modeling approaches could at least be augmented, if not founded on, the ability for engineers to construct new task-specific views by specifying dependencies on other engineers' views, and to control the integration of these views as the project progresses. Many of the requirements to enable the formalization of these project models discussed in this paper are in place today. As shown on the test cases, engineers already construct task-specific views to perform their design, planning and fabrication tasks. For example, the focus of this research has been on the construction and integration of geometric views consisting of geometric features. Many CAD programs

formalize such generic geometric views, such as layers, today. There is also a wealth of generic and task-specific geometric and other algorithms that transform source information into dependent information. What has been missing is a simple framework in which engineers can interactively relate these views and these algorithms into graphs of dependencies.

Providing engineers with tools to formally construct views from other views would provide a flexible, need-driven migration path from existing manual construction and integration of task-specific views to integrated and automated project models. Views and dependencies formalized on one project can be reused in subsequent projects. Additionally, these tools may support creative design processes, like those described in design theories such as Schon's (1984) "Conversation with a Medium", Gero's (1998) "Situatedness and Constructive Memory", and Smither's (1998) "Knowledge-Level Theory of Design." These theories frame design as a constructive process whereby engineers engage in new design acts based on current states of the design. Using the approach proposed in this paper, engineers may be able to iteratively and collaboratively construct and integrate graphs of views that help them understand and progress with the project from multiple perspectives, and engage in as-needed automated and integrated design and analysis.

## 6 References


Autodesk (2002a). "Building Information Modeling - White Paper", *Autodesk Incorporated* www.autodesk.com/buildinginformation

Autodesk (2003b). Autodesk Revit, www.autodesk.com.

Bentley, K., and Workman, B., (2003). "Does the Building Industry really need to start over?" *Bentley Systems Incorporated* http://www.bentley.de/about/neuigkeiten/BIM_WhitePaper.pdf, last accessed August 2003.

Abrahamson, S., Wallace, D., Senin, N., and Sferro, P. (2000). "Integrated Design in a Service Marketplace", Computer-Aided Design, 32, pp. 97-107.

Björk, B. C. (1987). "RATAS: A Proposed Finnish Building Product Model", Studies in Environmental Research, No. T6, Helsinki University of Technology, Otaniemi, Finland.

Bettig, B., Shah, J., and Summers, J. (2000). "Domain Independent Characterization of Parametric and Geometric Problems and Embodiment Design." Proceedings of DETC2000: Design Engineering Technical Conference, Sept., Baltimore, MD, 1 – 13.

CIM Steel Integration Standards, Release 2, URL: http://www.cis2.org.

Clayton, M., Teicholz, P., Fischer, M., Kunz, J. (1999). "Virtual Components Consisting of Form, Function, and Behavior." *Automation in Construction*, 8, 351-367.

Cutkosky, M.R., Englemore, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.R., Mark, W.S., Tenebaum, J.M., Weber. J.C.(1993). "PACT: a experiment in integrating concurrent engineering systems", IEEE Computer, 26:1, (January), 28-38.

Date, C. J., and Darwen H. (1993). *A Guide to the SQL Standard, Third Edition*, Addison-Wesley Publishing Company, Inc.

Eastman, C., Jeng, T-S., Assal, H., Cho, M., and Chase, S. (1995). *EDM-2 Reference Manual*, Center for Design and Computation, UCLA.

Eastman, C.M., Parker, D. S., and Jeng, T.S. (1997). "Managing the Integrity of Design Data Generated by Multiple Applications: The Theory and Practice of Patching", *Research in Engineering Design*, 9, pp. 125-145.

Eastman, C., and Jeng, T-S. (1999). "A Database Supporting Evolutionary Product Model Development for Design." *Automation in Construction*, 8 (3), 305-323.

Gero, J. S. (1998). "Conceptual designing as a sequence of situated acts" in I. Smith (ed.), *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp. 165-177.

Gielingh, W. (1988). *General AEC Reference Model*. ISO TC 184/SC4/WG1 doc. 3.2.2.1, TNO report BI, 88-150. .

Graphisoft (2003). *ArchiCAD 7*, http://www.graphisoft.com.

Hakim, M. M. and Garrett Jr., J. H. (1997). "An Object-Centered Approach for Modeling Engineering Design Products: Combining Description Logic and Object-Oriented Models." *Journal of AI in Engineering Design and Manufacturing*, Vol. 11, 187-198.

Haymaker, J.; Ackermann, E.; and Fischer, M. (2000). "Meaning Mediating Mechanism: A prototype for constructing and negotiating meaning in collaborative design," *6th International Conference on Artificial Intelligence in Design*; Kluwer Academic Publishers, Dordrecht, The Netherlands, 691-715.

Haymaker J., Suter, B., Fischer, M., and Kunz, J. (2003a). "The Perspective Approach: Enabling Engineers to Construct and Integrate Geometric Views and Generate an Evolving Project Model" Working Paper Nr 081, *Center For Integrated Facility Engineering*, Stanford University.

Haymaker J., Kunz J., Suter, B., and Fischer, M. (2003b). "Perspectors: Composable, Domain-Independent Reasoning Modules that Automatically Construct a Geometric Engineering View from Other Geometric Engineering Views" Working Paper Nr 082, *Center For Integrated Facility Engineering*, Stanford University.

Howard, H.C., Abdalla, J.A., and Phan, D.H.D. (1992). "Primitive-Composite Approach for Structural Data Modeling." Journal of Computing in Civil Engineering, ASCE, 6(1), 19-40.

IAI. (2003). "Industry Foundation Classes, Version 2.X." *International Alliance for Interoperability.* http://cic.vtt.fi/niai/technical/IFC_2X/index.htm .

Kam, C. and Fischer, M. (2002). Product Model & 4D CAD - Final Report, Technical Report Nr. 143, *Center For Integrated Facility Engineering*, Stanford University.

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

MacKellar, B. and Peckam, J. (1998). "Multiple Perspectives of Design Objects." *Artificial Intelligence in Design*, John Gero and Fay Sudweeks (eds.), Klewer Academic Publishers , 87-106.

Newton, R (2002). Implementing the Integrated Project Model in Stages: Robert Dalziel of Reid Architecture at the 2002 Teamwork Conference, http://www.msmonline.com/commentary. Last Accessed, August 15, 2003.

Post, N. (2002). "Movie of Job that Defies Description Is Worth More Than A Million Words", *Engineering News Record,* 248(13), 24.

Post, N. (2003). "Monster Wood Ceiling Crowns City of Angels' Music." *Engineering News Record,* 251(6), 30-33.

Progman Oy (2003). *MagiCAD*, http://www.progman.fi.

Rosenman, M. A. and Gero, J. S. (1996). "Modeling Multiple Views of Design Objects in a Collaborative CAD Environment." *CAD Special Issue on AI in Design*, 28 (3), 207-21.

Reina, P. (2003). "Staging, Props Could Not Keep 'Wavy' Roof's Original Tempo" *Engineering News Record,* 250(12), 30-33.

Sacks, R., Eastman, C.M., and Lee, G., (2003), 'Parametric 3D Modeling in Building Construction with Examples from Precast Concrete', Automation in Construction, Elsevier Science B.V., (in press).

Sacks, R., (2002). 'Integrated AEC Information Services using Object Methods and a Central Project Model', Computer-Aided Civil and Infrastructure Engineering, Blackwell Publishers, Boston USA & Oxford UK, Vol. 17 No.6 pp.449–456.

Schon, D (1983*). The Reflective Practictioner*. Harper Collins, New York, NY.

Serrano, D., and Gossard, D., (1987). "Constraint Management in Conceptual Design", Knowledge Based Expert Systems in Engineering, Planning and Design*, D* Sriram, and R.A. Adey (eds), *Computational Mechanics*, Southhampton.

Shah, J., and Mäntyla, M., (1995). *Parametric and Feature-Based CAD/CAM*, Wiley & Sons Inc., New York, NY.

Smithers, T. (1998). Towards a Knowledge Level Theory of Design Process, in J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '98, Kluwer, Dordrecht, 3–21.

Solibri Inc (2003). Solibri Model Checker, www.solibri.com.

Sriram, D. (2002). *Distributed and Integrated Collaborative Engineering Design*, Sarven Publishers.

STEP (2003). ISO 10303, Standard for the Exchange of Product Model Data.

Stiny, G, (1980). "Introduction to Shape and Shape Grammars." *Environment and Planning B: Planning and Design* 7, 343-351.

Stouffs, R. and Krishnamurti, R. (1997). "Sorts: A Concept for Representational Flexibility." *CAAD Futures*, ed. R. Junge, Kluwer Academic, Dordrecht, The Netherlands, 553-64.

Talukdar, S., Baerentzen, L. Goce, A. and derSouza, P. (1998). "Asynchronous Teams: Cooperation Schemes for Autonomous Agents." *Journal of Heuristics*, Vol. 4, 295 – 321.

Tow, D. R., and Harrison S. R.(2003). "Steel Star." *Modern Steel Construction*, American Institute of Steel Construction, Chicago, Ill. May Issue.

Turk, Z. (2001). "Phenomenological Foundations of Conceptual Product Modeling in Architecture, Engineering and Construction." *Artificial Intelligence in Engineering*,15 (2), 83-92.

Van Leeuwen, J. P. (1999). *Modeling Architectural Design Information by Features.* Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Van Nederveen, G.A., and Tolman, F.P. (1992). "Modelling multiple views on buildings." Automation in Construction, Vol. 1, 215-224.

Zamanian, M. K. and Pittman, J. H. (1999). "A Software Industry Perspective on AEC Information Models for Distributed Collaboration." *Automation in Construction* 8 (3), 237-48.