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

SUBMITTED: January 2004
REVISED: July 2004
PUBLISHED: October 2004 at http://www.itcon.org/2004/30/
EDITOR: Z. Turk

John Haymaker
Stanford University, Department of Civil and Environmental Engineering, Center for Integrated Facility Engineering
email: haymaker@stanford.edu

Martin Fischer
Stanford University, Department of Civil and Environmental Engineerin, Center for Integrated Facility Engineering
email: fischer@stanford.edu

John Kunz
Stanford University, Department of Civil and Environmental Engineering, Center for Integrated Facility Engineering
email: kunz@stanford.edu

Ben Suter
Stanford University, Department of Civil and Environmental Engineering, Center for Integrated Facility Engineering
email: bsuter@stanford.edu

SUMMARY: *To perform specific tasks, Architecture, Engineering, and Construction (AEC) professionals construct task-specific engineering views from information in other engineering views. Each view contains project information that is structured for an engineer's specific task; engineers are responsible for the information in their respective views. This paper presents industry test cases illustrating that due to the multidisciplinary, constructive, iterative, and unique character of AEC projects, these professionals have difficulty constructing and integrating their task-specific views today. We propose that engineers will benefit from expressive, generic, formal, and simple methods to iteratively construct a view by formalizing its dependency on other views, and control the integration of a graph of these views as the project progresses. This process maps well to the way AEC projects work today and would result in an integrated yet emerging project model consisting of a directed acyclic graph of task-specific views and dependencies. We define requirements for such an approach, and discuss that, while partially satisfying the requirements, existing approaches do not yet provide engineers with tools explicitly designed to enable them to iteratively construct views from other views, and control the integration of a graph of these views as the project progresses. To address these requirements, this paper introduces the Perspective Approach.*

KEY WORDS: *AEC, project model, dependency, integration, automation, task-specific views, 3D*

1. INTRODUCTION

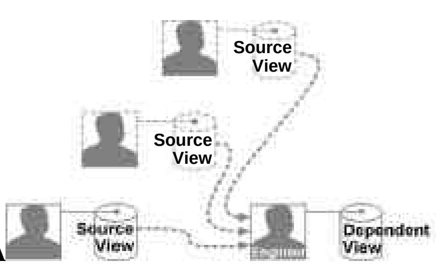
Engineers need integrated task-specific project representations, or views, to help them perform design, analysis, planning, fabrication, and assembly tasks. A view contains information about the project that is structured for their specific task. Per this definition, a view need not be generated from a model. These engineers could benefit from project-modeling approaches that provide them with such views more quickly and accurately than current practice and theory allows. 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:

- *Multidisciplinary*: 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 construction tasks.
- *Constructive*: Engineers construct their task-specific views from information in other engineers' views (Fig. 1 A). 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 (Fig. 1 B).
- *Iterative*: Engineers routinely modify 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, and must represent any implications of the modifications by manually 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 multidisciplinary 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.
- *Error-prone, time-consuming, and difficult*: Manually constructing and integrating dependent views from source views causes many problems on AEC projects today.

Traditionally, engineers have constructed and integrated views manually. For example, an engineer 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" tools) construct task-specific dependent geometric views in response to the information in source geometric views. However, manually constructing and integrating task-specific views, with pencil or CAD, remain difficult, error-prone, and time-consuming.

Project model approaches have extended CAD approaches to automate the construction and integration of 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 because they have not mapped well to the multidisciplinary, constructive, iterative, and unique characteristics of Architecture, Engineering, and Construction (AEC) projects.

A


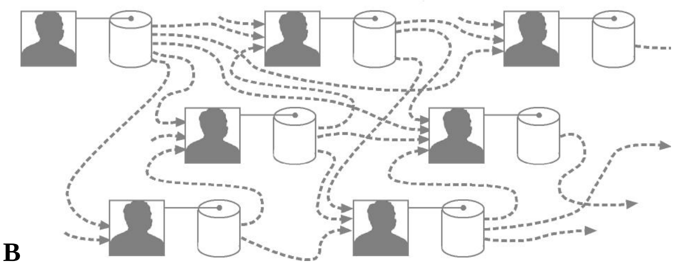
B


FIG. 1: The dependency between views. ***A.*** *An engineer uses task-specific reasoning to transform one or more source views, possibly 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 current practice does not explicitly formalize them. On AEC projects today, therefore, engineers must manually construct and integrate dependent views.* ***B.*** *Dependent views become source views for other dependent views. A directed graph of dependencies between views emerges.*

Based on the test cases, this paper argues that to address the difficulties engineers have on these projects, engineers may benefit from simple, formal, generic expressive methods to:

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

Given such methods, an integrated multidisciplinary project model emerges as a directed acyclic graph of views and dependencies between views. This is a simple but flexible and powerful idea that integrates product (views), organization (the engineers responsible for the views), and process (the dependency between views) as the project model evolves through time.

Many of the requirements for constructing and controlling graphs of views, as 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 today, for example layers in AutoCAD. 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 algorithms into graphs of dependencies.

This paper defines representation, reasoning, and management requirements for a project-modeling approach that provides engineers with the tools needed to construct views from other views and control an emerging project model of these views and dependencies. This paper also discusses existing AEC project-modeling approaches in terms of these requirements, and concludes that, while partially satisfying many of the requirements, existing approaches are not explicitly designed to enable engineers to iteratively construct views from other views and control the resultant graph of views and their dependencies.

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.

2. TEST CASES ILLUSTRATING THE MULTIDISCIPLINARY, CONSTRUCTIVE, ITERATIVE, AND UNIQUE CHARACTERISTICS OF AEC PROJECTS

This section presents three engineering test cases to illustrate the multidisciplinary, constructive, iterative, and unique character of AEC projects today. The test cases illustrate that to perform 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 could control the integration of their view with respect to these other views, they could address the time-consuming, error-prone, and difficult aspects of this process. This section concludes with representation, reasoning, and management requirements for testing the adequacy of alternative approaches to support engineers in constructing and integrating task-specific views in this way.

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 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. The design team used 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 design and construction processes. While the form of the building is unusual (see Fig. 2 A), 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 Fig. 2 B). For clarity these illustrations present just a main stairwell and public space that the project team called Element 2. Fig. 2 C 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 Fig. 2 D, to views such as those describing concrete slabs and steel framing (Fig. 2 E), to views detailing deck attachments that connect the steel beams to the concrete slabs (Fig. 2 F and 2 G). These views finally lead to the erection of the steel framing, deck attachments, metal deck, and concrete slab. Fig. 3 describes the approximate workflow executed by the project team. We represent this workflow as a graph of views and dependencies between views, which we also call a Narrative.

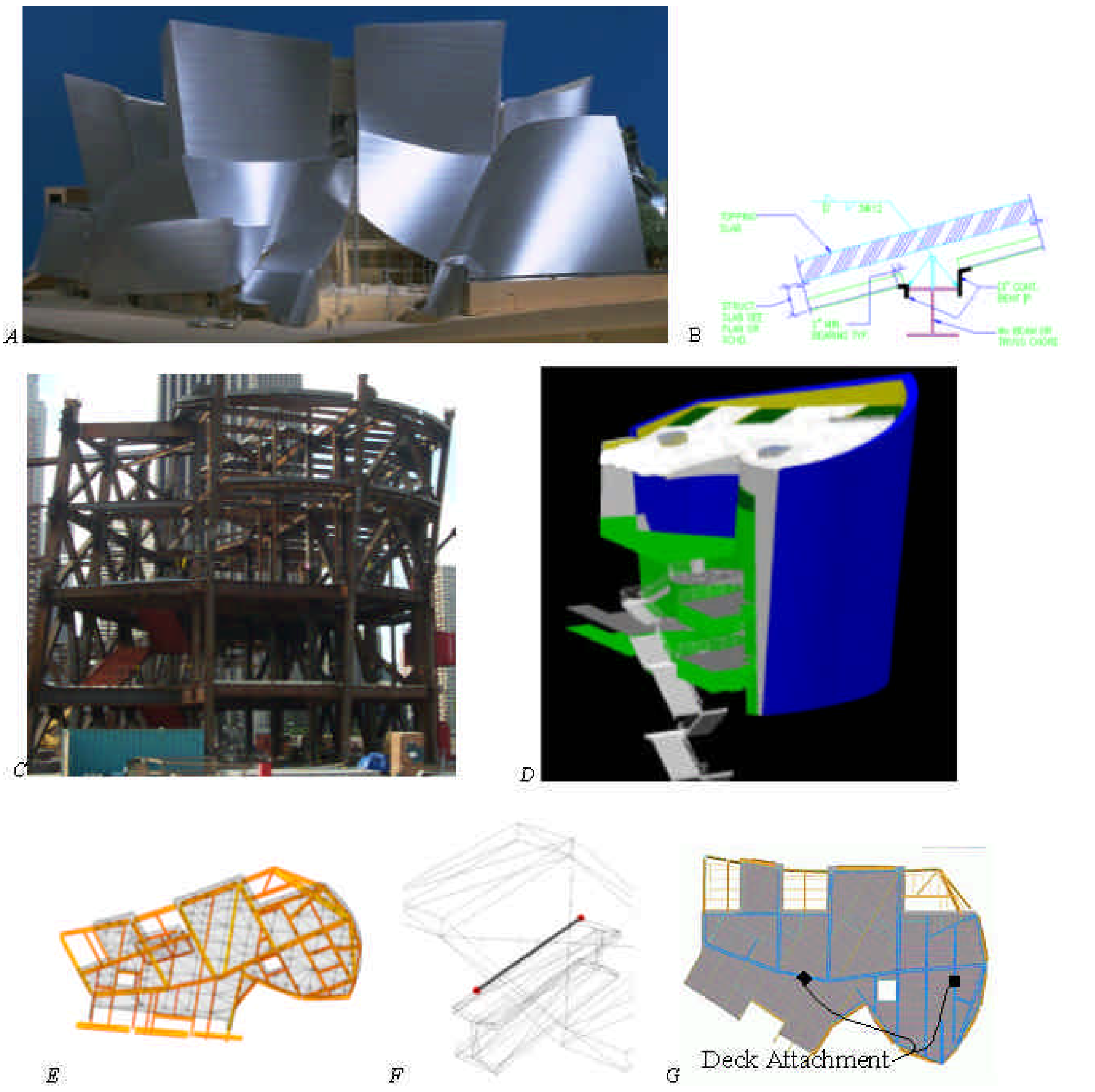


FIG. 2: Images of the WDCH for the deck attachment test case: ***A.*** *Physical model of the WDCH (image courtesy of Gehry Partners).* ***B.*** *Deck Attachment Detail view.* ***C.*** *Photograph of the Element 2 erected steel frame.* ***D.*** *Element 2 Exterior and Interior Visible Surface view.* ***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).*

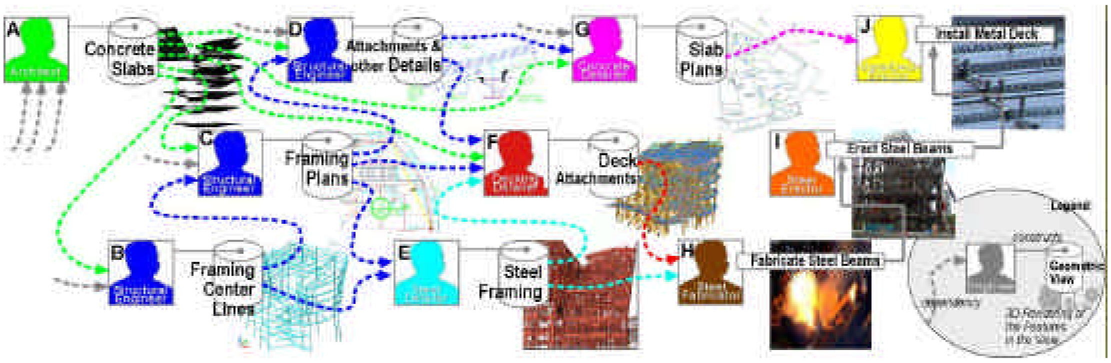


FIG. 3: The workflow for the design and construction team of the WDCH concrete and steel frame as described below.

A – G describe views and H – J describe construction processes that use these views (see Fig. 3):

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 the architect's visible surface view (see Fig. 2 D), 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 Fig. 2 B).

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 Fig. 2 G), consisting of a geometric line along the top edge of any beam where a Deck Attachment is required (see Fig. 2 F). 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.

During the design development and coordination, these engineers needed to iteratively modify their views. For example, Fig. 4 A 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 Fig. 3, thus requiring new and resized beams (Fig. 4 B) and new and revised deck attachment conditions (Fig. 4 C).

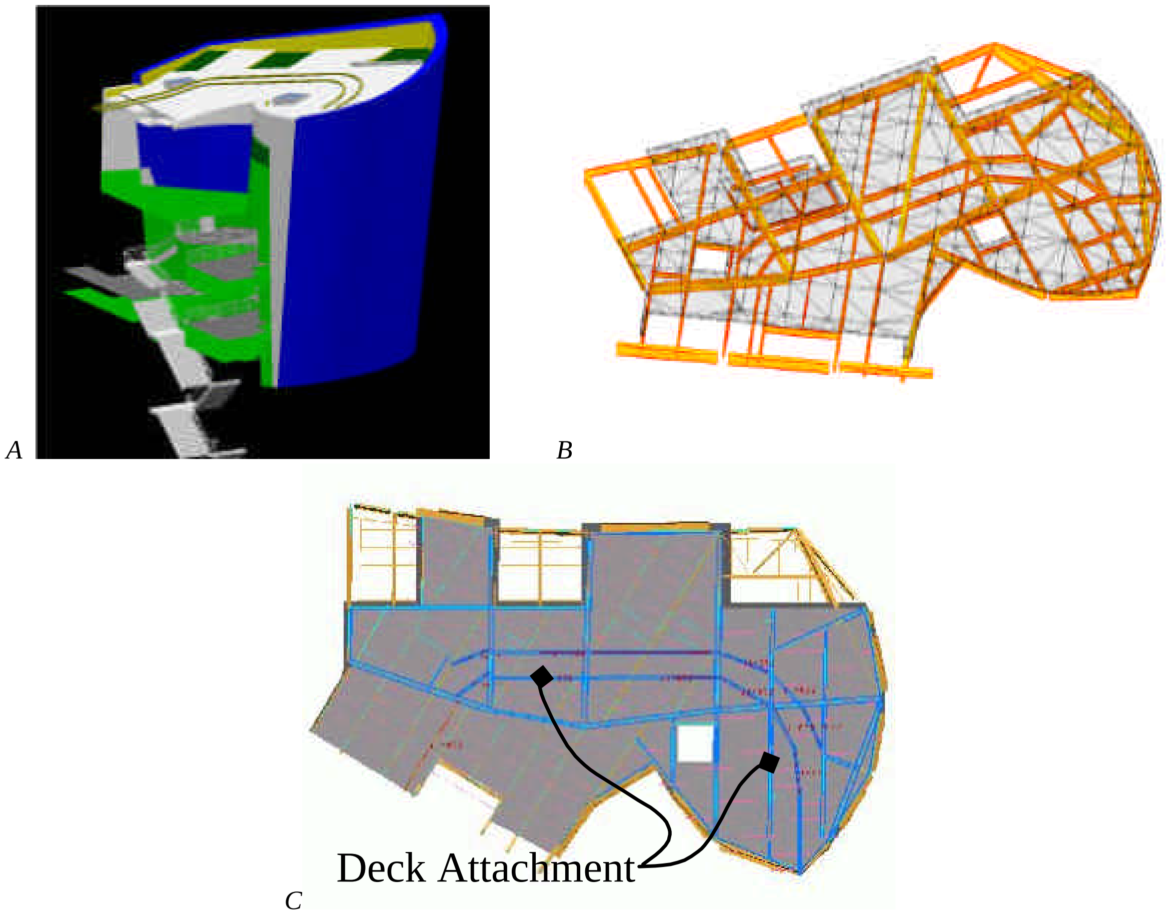


FIG. 4: A design modification in the deck attachment test case: A. Window washing tracks are added to the roof of Element 2. B. The tracks (and the crane) result in additional load that requires additional steel framing. C. The additional steel framing requires new deck attachments.

Fig. 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 because the correct information was not contained in the view. As a result the deck attachments required more expensive, time consuming, and error-prone field welding.

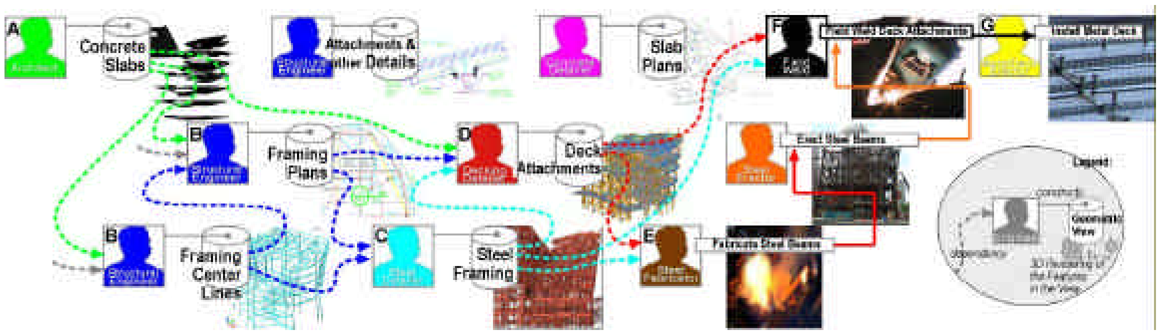


FIG. 5: The difficulties engineers had constructing and integrating the Deck Attachments view resulted in field welding of deck attachments. This diagram describes the propagation of the addition of window-washing tracks through the dependencies.

A. The architect added tracks for a window-washing crane to the design (see Fig. 5 A).

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 was not accurately updated.

E. If the deck attachments were 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, error-prone, and dangerous field welding of the deck attachments was required. Over $160,000 worth 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 installation of the metal deck and the work of subsequent trades.

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:

- *Multidisciplinary:* The architect, structural engineer, steel detailer, deck detailer, steel fabricator, etc. specialize 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 so that they can 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 cost 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 (Fig. 6 A) 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 (Fig. 6 B) 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 weighed in excess of 1,000 kg and was hung on four steel tube hangers; Each hanger was attached as close to each panel corner as possible (Fig. 6 C) 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 (Fig. 6 D). 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 limited available space.

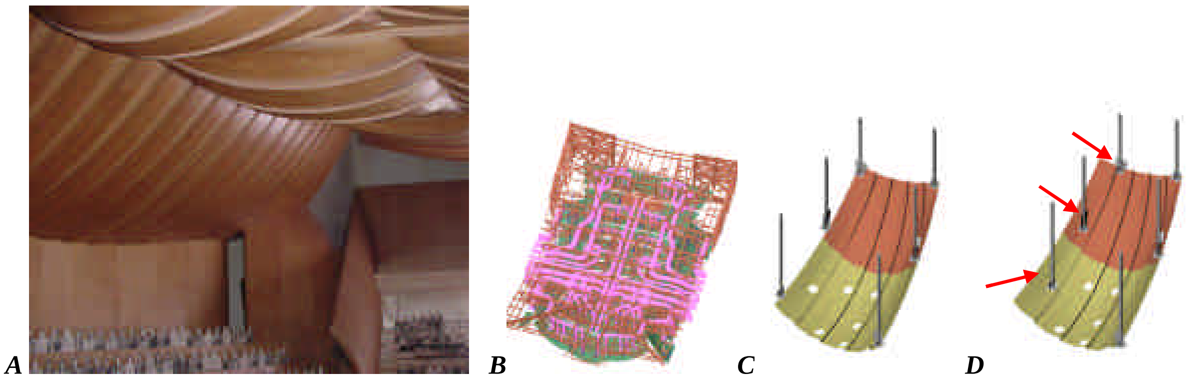


FIG. 6: Images of the WDCH for the cantilevered ceiling panel test case: ***A.*** *Interior view of the main hall ceiling (taken from a physical model built by Gehry Partners).* ***B.*** *The ceiling system from above: roof trusses and ductwork are overlaid above the ceiling panels and hangers.* ***C.*** *Two ceiling panels and six hangers with no cantilever conditions.* ***D.*** *The same two ceiling panels after a design change creates four cantilever conditions (pointed to with arrows: the middle hanger creates a cantilever condition on each panel).*

The engineers on this project did not construct and maintain an explicit 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, suggesting 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. Fig. 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 automatically and continuously highlighted. Wishing to minimize the number of panels with cantilever conditions, the team then uses this information in deciding which hangers to move when routing ducts.

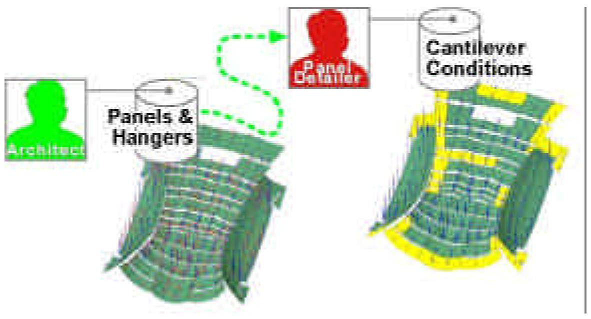


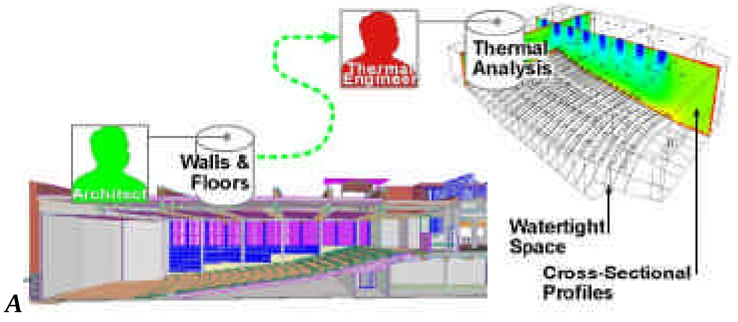
FIG. 7: A panel detailer constructs a Cantilever Conditions view from a view describing the locations of Ceiling Panels and Panel Hangers: The Cantilever Conditions view highlights panels that have cantilever conditions to assist in the routing of ducts.

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

- *Multidisciplinary:* The architect and panel detailer work for different organizations, and address task-specific engineering problems that require appropriate 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 iteratively 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 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 software developers.

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 support a thermal analysis (Fig. 8 A). Because of the configuration of the walls and floors in the auditorium (Fig. 8 B), the thermal analysis tool's usual extraction process could not construct the required Thermal Analysis view automatically from the architect's 3D 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. Ideally the thermal engineer would formalize how to construct the watertight space for this configuration of floors and walls, and would easily control the integration of this dependent view with the architect's source views (walls and floors).



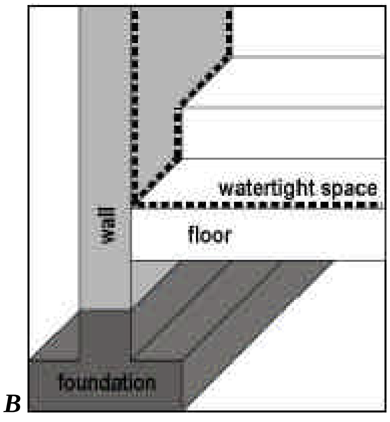


FIG. 8: Images of the thermal analysis test case. ***A.*** *The thermal engineer needs a watertight space in order to construct a Thermal Analysis view consisting of cross-section profiles of temperature stratification and air velocity values from the architect's view containing walls, floors, and other building components.* ***B.*** *The Architect's view of the walls and the floor of the auditorium frustrated the automatic extraction process for the thermal analysis because it did not explicitly describe the watertight space.*

The thermal analysis test case further reinforces the generality of the observation that AEC projects are multidisciplinary, 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, limiting the team's ability to go through multiple design and analysis cycles rapidly. 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 design, planning, and execution tasks. The information in these views is structured in a way that is suited to that task. Each engineer is contractually responsible for the information in his or her views. These engineers construct and integrate their views based on views produced by other engineers. As these views are iteratively modified, these engineers must integrate their views. Ideally engineers could automate the construction and integration of their views. One option is to have software application developers 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 been successful to date. The need for, and content of, task-specific views emerge as the design is developed by multiple disciplines. Our interpretation of the test cases is that multidisciplinary and unique AEC processes require exceptionally flexible representation and reasoning methods.

We investigate whether project-modeling approaches could at least be augmented by, if not founded on, methods to enable engineers to construct a task-specific view by formalizing its dependency on other views, and to control the integration of these views as the project progresses. For engineers to work in this way, such methods should be adequately:

- *Generic*: To apply across many different engineering disciplines.
- *Expressive*: To describe the many views and dependencies engineers need.
- *Formal*: To enable the methods to be implemented in a computer, in order to avoid the time-consuming, error-prone, and difficult manual construction and integration of views.
- *Simple*: To enable broad understanding, acceptance, and use by engineers.

From observation on the test cases, we propose the following methods, categorized as Representation, Reasoning, and Management methods:

Representation: Engineers could use an adequately generic, expressive, formal, and simple method to represent their task-specific views. It should be simple enough to be understood, yet also be adequately generic and expressive to describe a wide range of task-specific engineering concepts. The goal of this research is not to address what all the types of data in views could be. Other types of views than those described in the test cases, such as schedules and cost estimates, are also constructed and integrated from information in other views on AEC projects today. The goal is to formalize an adequately generic, expressive, formal, simple method to construct new dependent views from source views, and to control the integration of a graph of these views and dependencies. In order to test the concept, this research also requires adequately generic, expressive, simple data types to solve the test cases. In the test cases each view contains a collection of geometric features, for example those used to construct the Steel Framing, Concrete Slabs, Deck Attachments, Ceiling Panels, Panel Hangers, Cantilever Conditions, Walls, Seating Risers, and Watertight Space views. It would be useful to represent relationships between features in different views; for example, to represent the relationship between a deck attachment feature and the slab and beam features that it attaches.

Engineers could also use adequately expressive, generic, formal, and simple methods to represent dependency between views. 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 (DAG) of geometric views and their dependencies. Human engineers will manage the cycles in the dependencies. From observations on the test cases, the majority of the information dependencies are directed and managed in the context of views; therefore, this research investigates the simplicity of directed dependencies to help manage the complexity of multidisciplinary, constructive, iterative, and unique AEC processes. We formalize the following relationships and attributes to represent the dependency between a view and its source views:

- *Sources*: 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 becomes Not_Integrated.
- *Nature*: The reasoning method (automated or manual) that constructs the dependent view from source views. For example, the decking detailer uses reasoning to construct the Deck Attachments view from the Steel Framing and Concrete Slabs views.

Fig. 9 A diagrams this formalization of the dependency of a dependent view on source view(s). Fig. 9 B shows this representation method applied to the deck attachment test case.

Reasoning: Because of the unique nature of AEC processes, engineers (as opposed to software developers) could use adequately expressive, generic, formal, and simple methods to define the nature of the dependency. For example, Fig. 9 B shows reasoning called Find Deck Attachments that 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. Fig. 9 C shows that the method of formalizing the dependency between views should be able to be applied repeatedly. The reasoning can be manual (denoted by a human icon) or automated (denoted by a gears icon).

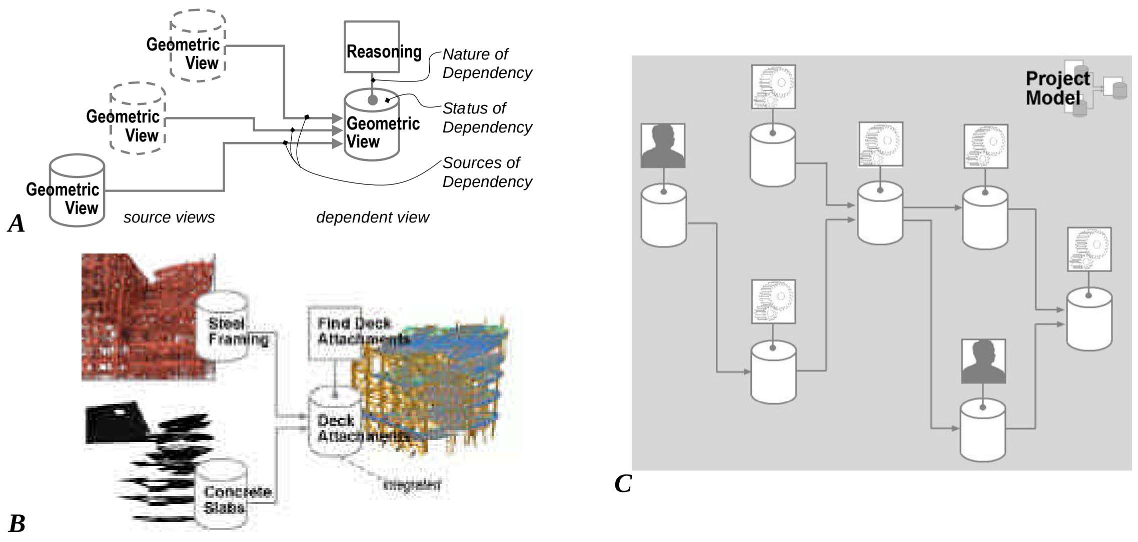


FIG. 9: Formalizing the dependency between views. ***A****. Formalizing the sources, nature, and status of the dependency of a dependent view on source views.* ***B****. Applying this formalism to the deck attachment test case.* ***C****. An example abstract project model emerges from the repeated application of the formalism described in A.*

Management: Engineers could use adequately expressive, generic, formal, and simple methods to manage 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:

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

For example, the engineer responsible for the Deck Attachments view should be able to easily construct the dependencies on the Steel Framing and Concrete Slabs views, receive notification 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.

2.5 Requirements to enable engineers to formalize the dependencies between views

The following summarizes our requirements for adequately generic, expressive, formal, and simple methods for engineers to construct views from views and control the integration of these views as the project progresses. We use these requirements in the next section to assess project-modeling approaches. Engineers need methods:

- Representation:
    1. To represent an engineer's task-specific views.
    2. To represent the sources, status, and nature of the dependency of a view on its source views.
- Reasoning:
    3. To automatically or manually construct information in a dependent view from information in source views
- Management:
    4. To iteratively construct (add, modify, and delete) instances of 1, 2 and 3.
    5. To control the iterative construction and integration of this graph of views and dependencies.

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

3. RELATED RESEARCH: PROJECT-MODELING APPROACHES

Many 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) have recognized the need for model evolution to allow engineers to construct new types of information and to define the sources, status, and nature of new types of dependencies between information. Given the project model requirements described in Section 2.4, this section discusses 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 (IFC) is 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 dependencies within this information. After an overview of approaches in this category, EDM-2 is discussed as it relates to the requirements.

Current representation and reasoning and management approaches are points of departure that satisfy many of the requirements of section 2.5, but they are not explicitly 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. Others construct and control dependencies between a federation of predefined views. In contrast, the Perspective Approach formalized in this research lets a project model emerge as a directed acyclic graph from the iterative construction and integration of an emerging graph of task-specific views and dependencies.

3.1 Representation approaches to integrated AEC project models

Some researchers approach integrated project-modeling by formalizing predefined 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 consisting of a subset 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 specialize to construct task-specific information and views. Representational approaches satisfy the representation requirements for a generic view, and might be used to formalize the sources and status of the dependency between views. However, representation approaches alone do not formalize the nature of the dependency between views, and they do not address the reasoning and management requirements.

For example, the Industry Foundation Classes (IFC) are an emerging industry standard for a project model schema that builds on many of the earlier representational approaches. The next section reviews the IFC to illustrate the extent to which representational schemas satisfy the requirements. The IFC are developed by the International Alliance for Interoperability, according to the IFC 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)*

Fig. 10 shows a part of the schema defined in IFC 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 formalize 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*.

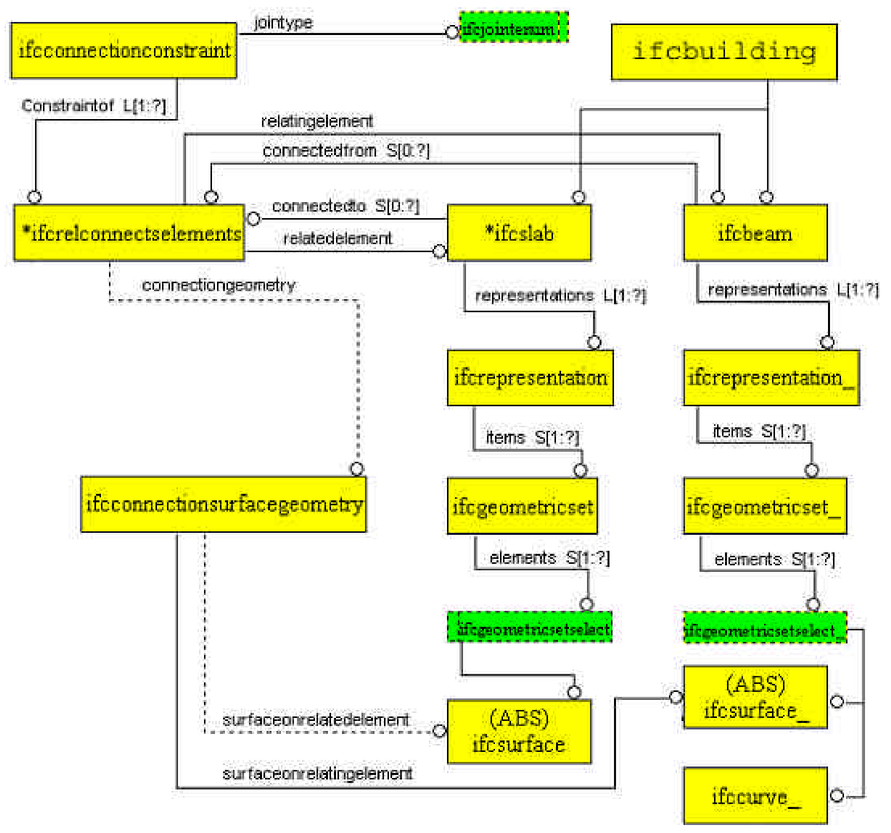


FIG. 10: Using the IFC to represent the connection relationship between a slab and beam. The IFC provides useful representation structures but often requires extensions to this representation (such as the concept of deck attachments). The IFC also lacks methods to construct and control dependencies within these representations.

Table 1 summarizes how the IFC relate to the requirements. The IFC 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 IFC 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 sources of the dependency of a Deck Attachment view on a Concrete Slabs and Steel Framing view. However, the IFC do not contain a formalization to represent the nature of the dependency, therefore requirement 2 is only partially satisfied.

The IFC make no commitment about reasoning or management processes; therefore, the IFC 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 IFC model.

Structured Query Language (SQL): Engineers can construct dependent views of an IFC 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 is not used broadly in AEC practice today to construct dependent geometric views. This is in part because these query languages have not provided geometric transformations that are formalized specifically to construct views from views, and in part because these engineers have lacked a simple, generic, formal, expressive framework that enables engineers to explicitly define these queries as dependencies between task-specific views and control the integration of these views and dependencies as the project progresses.

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 developers; the dependent views are not made available as source views for subsequent dependent views; and the sources and status of the dependency between views are not explicitly formalized or managed.

IFC 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 professionals to manually construct task-specific IFC views. These CAD programs provide feature construction tools that help engineers, for example, construct a beam, slab, or duct with a few mouse clicks. These tools lack a simple, generic, formal, expressive framework that enables engineers to explicitly define dependencies between these task-specific views and control the integration of these views and dependencies as the project progresses.

BSPro: BsPro (Granlund 2003) is a model server that stores the IFC model, and provides some tools for programmers to construct views of and write information to this model. BSPro is organized around the idea of the central model, but the sources and status of the dependency between views is not explicitly formalized or managed; rather, they formalize and manage dependencies between a central model and views.

These external applications partially satisfy requirement 3 of Section 2.5 because they enable engineers to construct many useful dependent views from information in the source view (the IFC model). However none of these tools enable 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.

Using representation approaches (with some extensions), engineers can manually construct many task-specific views and manually specify the sources and status of the dependency of information in these views on information in an IFC-based 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 novel 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 enable integrated task-specific project information, other researchers investigate reasoning and management approaches to augment representation approaches. Many in this reasoning and management category, such as Cutkoski et al (1993), Rosenman and Gero (1996), Eastman and Jeng (1999), Haymaker et al (2000), Sacks (2002), and Autodesk (2003a) construct and control dependencies between information in a central model. Others, such as Khedro and Genesereth (1994), Mackellar and Peckham (1998), Sriram (2002), and Bentley (2003) construct and control dependencies between a federation of task-specific views. These systems neither formalize tools to iteratively construct views from views, nor tools to control a graph of these views and their dependencies.

Some report success constructing and controlling dependencies within the context of single domains or between two domains (Sacks et al 2003, Tow and Harrison 2003, Reina 2003, Mora et al 2004). However, 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 multidisciplinary, constructive, iterative, and unique nature of AEC projects.

As an example of the reasoning and management approaches, this section compares the Engineering Data Model, also known as EDM-2 (Eastman and Jeng 1999), 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. However, it does not provide the representation, reasoning, and management concepts to provide engineers with the guidance needed to iteratively construct and control a project model consisting of a graph of views and dependencies between views, and thus does not fully meet our requirements.

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). Fig. 11 shows how Eastman and Jeng diagram this approach.

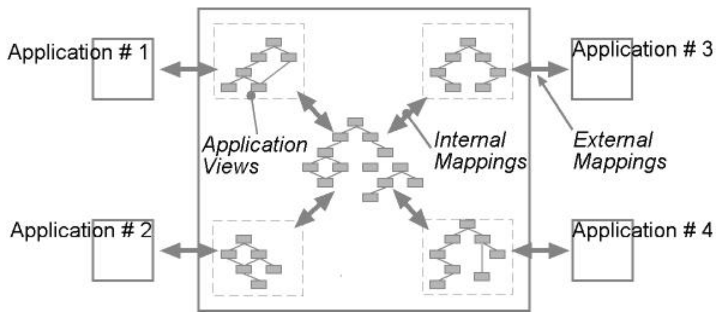


FIG. 11: EDM-2: a product model based on multiple application views and a central model consisting of Design Entities (gray rectangles). Image is redrawn from Eastman and Jeng (1999).

The test cases illustrate that engineers could benefit from adequately expressive, generic, formal, and simple methods to represent 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 representation schema with which to construct a view, and therefore satisfies requirement 1. The test cases also illustrate the need for methods to formalize the sources, status, and nature of the dependency between views. EDM-2 formalizes Constraints, which are used to formalize the sources 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 sources, status, and nature of the dependencies of a DE on other Des; therefore, EDM-2 only partially satisfies requirement 2.

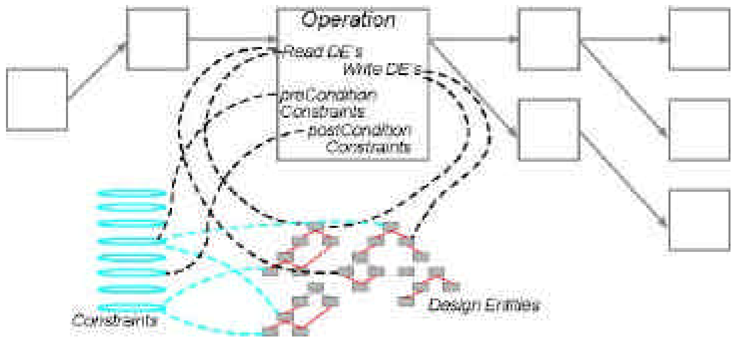


FIG. 12: EDM-2 formalizes precedence dependencies between Operations that iteratively access and construct a central model of Design Entities. Constraints control these Operations. (Adapted from Eastman et al (1997)).

The test cases also illustrate that engineers require reasoning methods to construct new task-specific views by formalizing the nature of their dependency on other task-specific views. Fig. 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, and currently system programmers are required to construct the dependencies and the views in EDM-2; therefore, requirement 3 is only partially satisfied.

Finally, the test cases also illustrate that engineers require management methods to construct and control this graph of views and dependencies. The integration system described 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 to encode the nature of the dependencies. This architecture is similar to model-view-controller architectures (Reenskaug 1979) employed in other systems (Turk 1994). While EDM-2 controls a project model of views organized around a central model, EDM-2 does not 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.

EDM-2 is a database language that contains mapping techniques to integrate information in views that are organized around a central building model. Who defines, 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 multidisciplinary 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.

3.3 Other reasoning and management approaches

Other researchers also investigate reasoning and management approaches. Shape Grammars (Stiny 1980) provide a computational approach to the generation of designs based on rules that find and replace geometric information in a model. 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). 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 in a central model. Exemplars and their algorithms (Summers et al 2004) facilitate four basic design tasks on a central model: pattern matching, property extraction, design validation and change propagation. Both use reasoning to transform information from one view into information in other views. DOME (Abrahamson et al 2000) provides a publish, subscribe, and synthesize 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. Parametric approaches match the spirit of the requirements for AEC project-modeling derived from the test cases. However, they do not contain adequately generic, expressive, formal, and simple methods to explicitly guide AEC engineers in constructing a view from information in other views and controlling the integration of a graph of these views as the project progresses.

3.4 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 maps closely to the way AEC engineers work today. 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 IFC formalize many representational concepts, although they do not explicitly formalize the sources, 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, multidisciplinary project model consisting of a graph of views and dependencies.

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

4. THE PERSPECTIVE APPROACH

The *Perspective Approach* specifically formalizes methods to construct views from other views and control the integration of a directed acyclic graph of views and dependencies as the project progresses. In the Perspective Approach (Fig. 13 A) a task-specific view is called a *Perspective*. As shown in Table 1, 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 sources, 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 assist an engineer to construct this information manually. Because both a Perspector's input and output are Perspectives, the approach can be applied in a modular fashion. 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, which we call a Narrative, develops over time and is represented formally as a directed acyclic graph of dependencies between Perspectives, where the nature of these dependencies is represented by Perspectors. We call a graph of Perspectives and Perspectors a Narrative. 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, Fig. 13 B 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* that contain geometric data types, and relationships to other Features. 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). Fig. 13 C shows that an engineer (also the first author of the paper) can use the Perspective Approach and reusable geometric Perspectors to compose a Narrative to formalize the Find Deck Attachments Perspector. The Find Deck Attachments Perspector can then be defined by subsuming this Narrative.

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 Perspectives. The paper provides evidence that engineers may be able to select from a reasonably small number of predefined, reusable geometric Perspectors and compose and customize them into Narratives that specify automatic or manual transformations of many source geometric Perspectives into dependent geometric Perspectives more quickly, accurately, and less expensively than current practice allows.

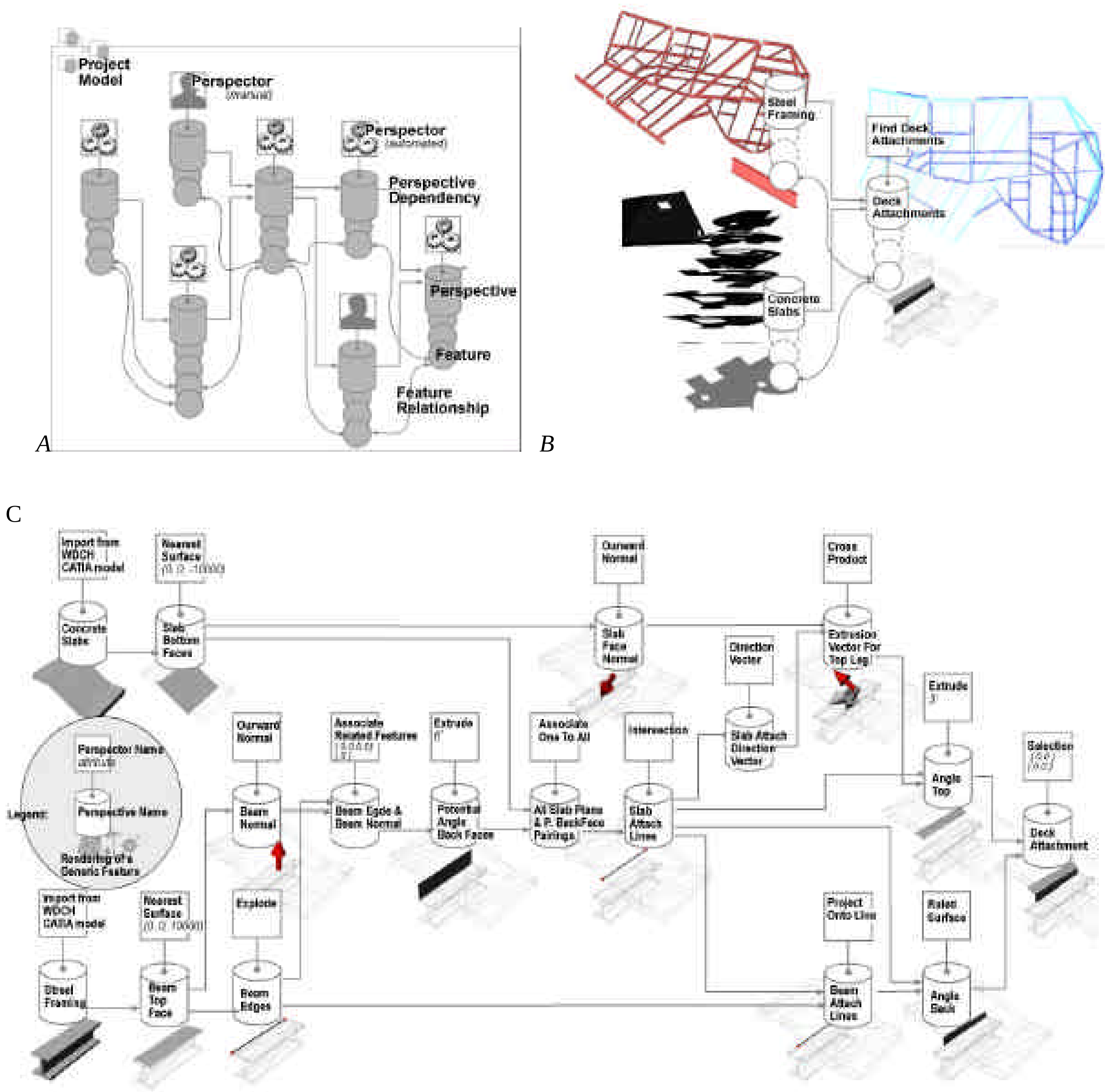


Fig. 13: The Perspective Approach enables an engineer to construct and control the integration of a dependent Perspective from changing source Perspectives. ***A.*** *A geometric Perspective is a task-specific view containing Features that use geometry (not shown) and relationships to other Features to describe task-specific engineering concepts. A Perspector is a reasoning mechanism that analyzes the Features in source Perspectives to construct Features in the dependent Perspective. A Perspector can be automated (shown as gears) or manual (shown as a human).* ***B.*** *Applying the Perspective Approach to the Deck Attachment test case: The Find Deck Attachments Perspector analyzes the slab Features in the Slabs Perspective (produced by the Architect) and the steel framing Features in the Steel Framing Perspective (produced by the Steel Detailer) to automatically construct deck attachment Features in the Deck Attachments Perspective. The Find Deck Attachments Perspector also relates deck attachment Features with their associated slab and steel-framing Features.* ***C.*** *Composing a directed acyclic graph of low-level Perspectors and Perspectives, or a Narrative, 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. A rendering of a typical feature is shown under each Perspective. An engineer can then subsume this Narrative into the Find Deck Attachments Perspector shown in B. These Perspectors are described in detail in Haymaker et al (2003b).*

Representation:	IFC	EDM-2	Perpsective Approach
1. A generic engineering view	Ifcgroup,Ifcobject, Ifcsurface	Design Entity, Composition	Perspective, Feature
2. Source, nature, and status of the dependency between views	▨ Ifcconstraint	▨ Constraints, Operations, Maps	Source Perspectives, Perspector, Status
Reasoning:			
3. Construct dependent view from source views	▨ External Applications	▨ Operations, Maps	Perspector
Management:			
4. Construct views and dependencies on other views	■	▨ System Developers	Management Processes
5. Control the integration of new and modified views	■	▨ Operations	Management Processes

Legend	
□	Satisfies
▨	Partially satsifies
■	Does not satisfy

Table 1: Comparing the Industry Foundation Classes (IFC), Engineering Data Model-2 (EDM-2), and the Perspective Approach to the requirements of Section 2.5.

Evidence for the power and generality of the Perspective Approach is provided in these papers by implementing a software prototype on the two test cases presented in this paper from the design and construction of the Walt Disney Concert Hall. The prototype was implemented in the Java programming language to run on a single machine. Geometry for the test cases was exported from CATIA using the Virtual Reality Modeling Language (VRML) and transformed into an internal format where each VRML file corresponds to a Perspective, and each VRML Group was transformed into a Feature in the Perspective. A Feature contains a name, a collection of surfaces, lines and points, and relationships to other Features. Fig. 14 shows the user interface to the prototype. An engineer can select existing Perspectors, or program new Perspectors, and compose and control the integration of the Narrative of Perspectives (and their associated Perspectors) in the bottom left window. The engineer can view a 3D visualization of each Perspective in the upper window, and can view a tree-based view of Perspectives and Features in the bottom right window.

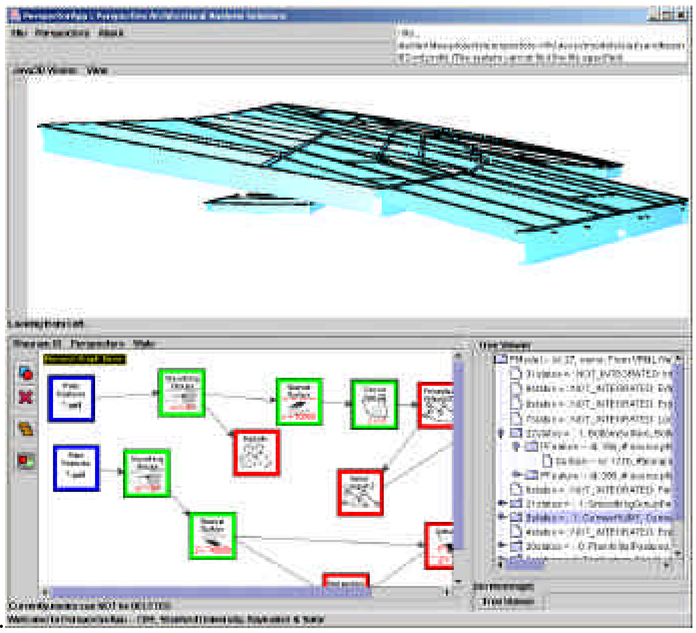


Figure 14: A prototype of the Perspective approach applied to the deck attachment test case. The 3D view in the upper window of the application shows a Steel Framing Perspective with a Beam Edges Perspective overlaid. An Engineer (the first author of the paper) assembled a graph of Perspectors (and their associated Perspectives) to construct new dependent Perspectives from source Perspectives. The graph of Perspectives and Perspectors (also called a Narrative) is constructed and controlled in the lower left window. Integration status is displayed using color (red for Not Integrated, green for Integrated).

5. DISCUSSION

The test cases presented in this paper demonstrate that engineers today have difficulty constructing and integrating geometric views on multidisciplinary, constructive, iterative, and unique AEC projects, even using state-of-the-art tools like CATIA. To address these difficulties, engineers could benefit from adequately expressive, generic, formal, and simple methods to iteratively construct views from views, and control the dependencies between these views as the project progresses. Representation approaches for AEC project-modeling such as the IFC define many useful ways to represent task-specific concepts, and that reasoning and management approaches for AEC project-modeling, such as EDM-2, formalize the dependencies between concepts, or between a central model and views. However, these approaches do not explicitly formalize methods that enable engineers to construct a new task-specific view by specifying its dependency on other engineers' views, nor do they formalize methods that enable engineers to control the integration of a graph of these views as the project progresses.

Formalizing a project model as a Narrative of views and dependencies has limitations. There are likely problems to which a central model with derived views such as those proposed by the IFC and EDM-2 is a more appropriate solution; for example, in the design and construction of less unique structures such as car parks or typical office buildings (House 2004). Additionally, the directed nature of the graphs does not provide formal support for cases where there are cycles in the dependencies. For example, in the test cases 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 of views and their dependencies, allowing engineers to manually manage the cycles.

Our research method has limitations. We qualitatively assessed project-modeling approaches in terms of their formalization of expressive, generic, and simple methods to construct and control a graph of views and dependencies. Future work is needed to quantify and validate various approaches against these metrics. In Haymaker 2003a,b we provide evidence that the representation, reasoning, and management formalized in the Perspective Approach are adequately simple, generic, formal, and expressive to be used to automate and integrate the task-specific views of the different disciplines on the test cases. More test cases are needed to further develop and classify the representation, reasoning, and management methods proposed in this paper.

Providing engineers with tools to formally construct views from other views provides a flexible, on-demand migration path from existing manual construction and integration of task-specific views to automatically constructed and integrated views. Narratives of views and dependencies formalized on one project can be modified and reused in subsequent projects. Design is a constructive process whereby engineers engage in new design tasks based on current states of the design (Schon 1984, Gero 1998, Smithers 1998). Using the approach proposed in this paper, engineers are able to iteratively and collaboratively construct and integrate Narratives that help them understand and progress with the project from multiple perspectives, and engage in as-needed automated and integrated design and analysis.

6. ACKNOWLEDGEMENT

Several organizations and individuals have significantly contributed to this research, including MA. Mortenson Company, Gehry Partners, Walt Disney Imagineering, Columbia Showcase, Martin Brothers/Marcowall Inc., Consolidated Contractors Company, George Haymaker Jr., Jean-Claude Latombe, and Barbara Tversky.

7. REFERENCES

Autodesk (2003a). "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, 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.

CIM Steel Integration Standards, Release 2, 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: An 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, 125-45.

Eastman, C., and Jeng, T. S. (1999). “A Database Supporting Evolutionary Product Model Development for Design,” *Automation in Construction*, 8 (3), 305-23.

Gero, J. S. (1998). “Conceptual Designing as a Sequence of Situated Acts,” *Artificial Intelligence in Structural Engineering*, Smith, I. (ed.), Springer, Berlin, pp. 165-77.

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, 187-98.

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.

House, M. (2004). “Technology and Object-Oriented Design Tools,” Presentation at AEC Technology Strategies 2004, Las Vegas, Nevada, http://events.zweigwhite.com/technology2004/program.htm.

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. and Genesereth, M. R. (1994). “The Federation Architecture for Interoperable Agent-Based Concurrent Engineering Systems,” *International Journal on Concurrent Engineering, Research and Applications*, 2, 125-31.

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

Mora, R., Rivard, H., and Bedard C. (2004). “Computer-Aided Conceptual Design Of Building Structures,” *Design, Computing and Cognition*, John Gero (ed.), Kluwer Academic Publishers.

Newton, R. (2002). “Implementing the Integrated Project Model in Stages: Robert Dalziel of Reid Architecture,” 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), 30-3.

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.

Reenskaug T. (1979). “Models – Views – Controllers,” Technical Note, Xerox Parc, http://heim.ifi.uio.no/~trygver/mvc/index.html.

Reina, P. (2003). “Staging, Props Could Not Keep ‘Wavy’ Roof's Original Tempo,” *Engineering News Record,* 250(12), 30-3\3.

Sacks, R., Eastman, C.M., and Lee, G. (2004). “Parametric 3D Modeling in Building Construction with Examples from Precast Concrete,” *Automation in Construction*, 13(3), 291-312.

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 and Oxford UK, Vol. 17 No.6, 449-56.

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,” *Artificial Intelligence in Design ‘98*, Gero, J. S. and Sudweeks, F. (eds), 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, 7*, 343-51.

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

Summers J., Bettig, B, Shah, J. (2004). “The Design Exemplar: A New Data Structure for Embodiment Design Automation,” *ASME Journal of Mechanical Design*, 126 (13).

Talukdar, S., Baerentzen, L. Goce, A. and derSouza, P. (1998). “Asynchronous Teams: Cooperation Schemes for Autonomous Agents,” *Journal of Heuristics* (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, 55-64.

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

Turk, Z. (1994) “Construction Design Document Management Schema and Prototype,” *International Journal of Construction Information Technology*, 2 (4), 63-80.

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). “Modeling Multiple Views on Buildings,” *Automation in Construction*, 1, 215-24.

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.