CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

The Perspective Approach: Enabling Engineers to Construct and Integrate Geometric Views to Generate an Evolving Project Model

By

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

CIFE Working Paper #WP081
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

THE PERSPECTIVE APPROACH: ENABLING ENGINEERS TO CONSTRUCT AND INTEGRATE GEOMETRIC VIEWS AND GENERATE AN EVOLVING PROJECT MODEL

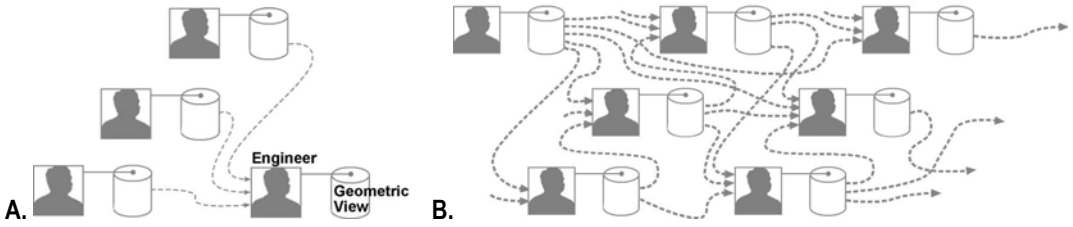
JOHN HAYMAKER, BEN SUTER, MARTIN FISCHER, JOHN KUNZ
Stanford University, Department of Civil and Environmental Engineering
Center for Integrated Facility Engineering, Building 550, Room 553H,
Stanford, CA 94305
haymaker@stanford.edu, bsuter@stanford.edu, fischer@stanford.edu, kunz@stanford.edu

Abstract

Enabling engineers to construct integrated and task-specific views of an evolving AEC project is an important and unresolved issue in both practice and research. Many current project-modeling approaches construct and integrate a predefined central model from which task-specific views are constructed. Others construct and integrate a "federation" of predefined task-specific views. AEC processes have been slow to adopt these approaches. This paper formalizes the Perspective Approach, in which engineers from multiple disciplines iteratively construct geometric engineering views, called Perspectives, from information in other geometric Perspectives and control the integration of this multi-disciplinary, evolving Project Model as the project progresses. The paper describes an implementation of this approach on two test cases from the design and construction of the Walt Disney Concert Hall, showing how the Perspective Approach would enable engineers from multiple disciplines to construct and integrate task-specific views of an evolving AEC project more quickly, accurately, and simply than current practice allows.

1 Introduction

Architects, engineers, contractors, fabricators, and other AEC professionals use task-specific views to design, plan, and execute AEC projects. Today, they often construct (in other words add, modify or delete) information in these views based on information in other engineers' views (Figure 1A). These *dependent* views often become *source* views of other dependent views: A network of dependencies between distributed, task-specific views emerges as the design progresses (Figure 1B). When source views are modified, dependent views often must be integrated. Unfortunately, in practice the dependency network is implicit, and therefore unavailable for inspection, review, reuse or management. Based on industry test cases, Haymaker et al (2003a) illustrate the *multi-disciplinary*, *constructive*, *iterative*, and *unique* characteristics of AEC projects and suggest that engineers on these projects could benefit from a simple approach to formally construct views from information in other engineering views, and control the integration of these views. Such an approach would enable an integrated project model to emerge as a directed acyclic graph (d.a.g.)[1] of task-specific views and their dependencies. That paper develops requirements for an approach that will formally support this process and reviews prior work in terms of these requirements. That paper concludes that, while important as points of departure, the prior work in AEC project modeling has not formalized an approach that provides AEC engineers with the guidance to easily construct a task specific view by formalizing its dependencies on other engineering views and control a project model that emerges from the iterative application of this method.



A. *Engineers construct and integrate views from other views.* ***B****. The process in A occurs iteratively: A project model emerges as a graph of directed dependencies between views. The dependencies are drawn as dashed lines because these dependencies are mostly implicit on AEC projects today.*

Figure 1:The dependency between views.

To satisfy the requirements, this paper describes the Perspective Approach. The Perspective Approach enables an engineer to construct a task-specific dependent engineering view, called a Perspective, by formalizing its dependency on source Perspectives. A Project Model emerges as a directed acyclic graph of task-specific Perspectives and dependencies between Perspectives. The Perspective Approach also formalizes simple Management Processes that assist engineers to construct Perspectives, and control the integration of Perspectives with respect to the Perspectives on which they depend.

To formalize its dependencies, a Perspective contains:

1. Relationships to its source Perspectives to describe the *existence* of the dependency on its sources.

[1] A directed acyclic graph is a special kind of network in which the arcs are directed and there are no cycles. In the Project Model presented in this paper, the nodes are the views, and the arcs are the dependencies between views.

2. A representation of the *status* of the integration of this Perspective with respect to its source Perspectives.
3. A relationship to a reasoning mechanism called a *Perspector* that formalizes the *nature* of the dependency.

A Perspector constructs information in the dependent Perspective from information in its source Perspectives according to an algorithm. A Perspector can be completely automated, or it can simply provide CAD tools for an engineer to construct the information manually. Formalizing a project model as Perspectives and dependencies results in an easily extensible project model that evolves naturally as engineers from multiple disciplines construct and integrate task-specific Perspectives of other task-specific Perspectives.

Perspectives are generic views that engineers can use to contain many different data types to describe many different types of project views (such as schedule data, cost estimates, etc.). This paper investigates the construction and integration of geometric Perspectives. Geometric Perspectives are like most CAD layers that engineers construct today; they contain *Features* that describe the project for a specific task. Features can be named and can contain geometric data types and relationships to other Features in other Perspectives. In these Perspectives, several Features interact in 3D space. Engineering conditions of interest to a particular task emerge between some, but not all of these concepts.

This paper presents a prototype that implements the Perspective Approach using geometric Perspectives on two test cases from a recent, state-of-the-art project called the Walt Disney Concert Hall (WDCH), showing retrospectively how the approach can be used to construct and control the integration of these task-specific geometric views more quickly and accurately than was possible on the WDCH.

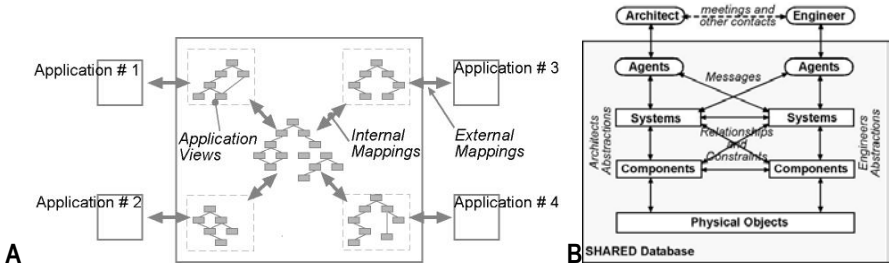
This paper contains two contributions to the theory of AEC project modeling. The first is the specification of a generic view called a geometric Perspective that formalizes the existence, status, and nature of its dependency on other geometric Perspectives. The second is the specification of simple Management Processes that formalize how engineers can iteratively construct and control Perspectives and their dependencies.

2 Overview of Integrated Project Modeling Approaches

Traditionally, AEC engineers have constructed and integrated views manually and implicitly; for example, overlaying transparent drawings to assist in constructing and integrating a project's geometry in two dimensions. 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 information in a dependent geometric view in response to 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. In today's practice, the existence, status, and nature of the dependencies between project information is often not formally represented.

Project model approaches have extended CAD approaches to address the need to integrate project information. Representational approaches (Björk 1987, Gielingh 1988, STEP 2003, IAI 2003) specify a schema consisting of formal ways to represent typical AEC information, such as 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 construct task-specific views by selecting a subset of this model. These approaches can contain constraints to formalize the existence and status of a dependency between information. However these constraints do not explicitly formalize the nature of the dependency when information in one part of the model requires construction of information in another part of the model.

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 and Workman 2003) are recognizing the need to formalize reasoning and management approaches that support model evolution for AEC projects. Some of these approaches (i.e., Eastman and Jeng 1999, Haymaker et al 2000, Sacks et al 2003, Autodesk 2003) develop reasoning and management that constructs and controls dependencies of information in a pre-defined central model (Figure 2A). Others (Khedro and Genesereth 1994, Rosenman and Gero 1996, Mackellar and Peckham 1998, Sriram 2002, Bentley 2003) develop similar reasoning and management approaches that construct and control dependencies between information in a federation of predefined task-specific views (Figure 2B). In both central and federated model approaches, system programmers are generally required to construct the dependencies.



***A.** EDM-2 manages dependencies between views and a central model (Eastman and Jeng1999). **B.** SHARED manages dependencies between a federation of views (Sriram 2002).*

Figure 2: Integrated project model approaches.

Parametric approaches (Shah and Mäntyla 1995) are reasoning and management approaches that provide engineers with tools to define dependencies between information. Most parametric approaches commonly use a graph (Serrano and Gossard 1987), often called a history tree, to structure the dependencies between information in a central model. Commercially available parametric modelers for the mechanical engineering industry such as CATIA (Dassault, 2003) provide tools to assist engineers to construct 2D sketches from which 3D features are parametrically constructed, and to specify how to parametrically position 3D features in relation to other 3D features. 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 Xsteel (Tekla 2003), Revit (Autodesk 2003), and Microstation TriForma (Bentley 2003b). While success is reported within the context of single domains (Sacks 2003), parametric approaches are not yet widely used in the AEC industry to integrate the work of multiple disciplines.

In the mechanical engineering domain, Abrahmason et al (2000) formalize a service marketplace, called DOME, in which engineers can "publish and interrelate services". In DOME, engineers integrate information in their views with information in other engineering views. The Perspective Approach presented in this paper shares a similar goal, specializing this idea into a formalism that emphasizes the multi-disciplinary, constructive, iterative and unique characteristics of AEC projects by guiding engineers to explicitly construct engineering views from other engineering views and to control the emerging project model.

3 The Opportunity to Formalize a Project Model as a Directed Acyclic Graph of Geometric Views and Dependencies

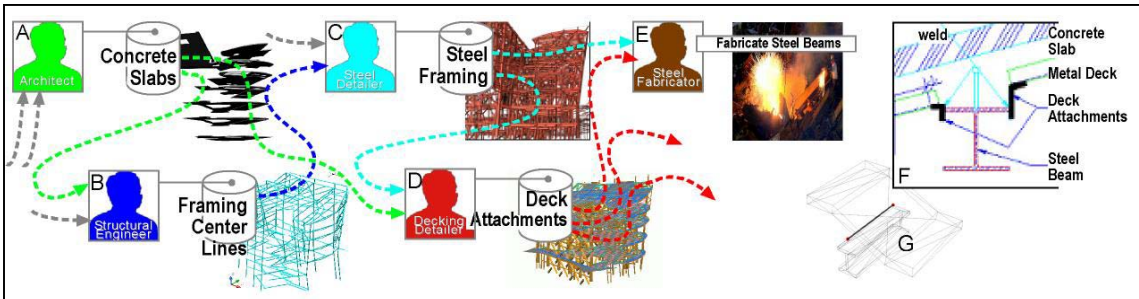
Haymaker et al (2003a) suggest from observations on test cases that engineers could benefit from a formal yet intuitive way to construct and control task-specific views by formalizing their dependencies on other views, developed requirements for a project model approach that would support engineers in this process, discussed prior work as they relate to our requirements, and concluded that while points of departure, current project modeling approaches do not enable engineers to easily construct new task-specific views of other engineers' views, or to control an evolving project model of views and their dependencies. This section summarizes these test cases, observations, and requirements, adding to them the observation that AEC engineers today often construct geometric views consisting of geometric features, and that these features contain implicit relationships to features in other views.

3.1 Test cases: Illustrating the multi-disciplinary, constructive, iterative, and unique characteristics of AEC projects

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 WDCH. As the project progressed they subcontracted work to various engineering firms and subcontractors that specialize in specific tasks of the building lifecycle. 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. These engineers then modified their views to reflect the current state of the project. This section briefly describes two test cases from this project.

3.1.1 The deck attachment test case

Figure 3 describes the approximate workflow observed on the design and construction of the steel frame of the WDCH project. The test case focuses on the design and installation of the deck attachments to illustrate the opportunity to formalize a project model as a directed acyclic graph of geometric views and dependencies. The steps A-E represent the flow of product information through the organization as these actors perform some design process on different aspects of the design.



A. *The architect constructed and integrated a Concrete Slabs view containing features that use a surface to describe the boundary of each concrete slab on the project. The architect constructed this view using information in several other project views (not shown).*

B. *The structural engineer constructed a Framing Center Lines view containing features that use a line to describe the centerline of each steel member required for the frame of the building. The structural engineer constructed this view using information in the Concrete Slabs view and other views.*

C. *The steel detailer constructed a Steel Framing view containing features that use a surface to describe the boundary of each steel member and other features to describe connections to other steel members. The steel detailer constructed this view, using information in the Framing Center Lines view and other views not shown.*

D. *The metal decking detailer constructed a Deck Attachments view containing features that use a line to describe where to install attachments to connect the metal decking for concrete floor slabs to the structural beams (see Figure **F**). The metal decking detailer constructed this view using information in the Concrete Slabs and Steel Framing views, by drawing a line along the edge of each beam where an attachment is required (see Figure **G**).*

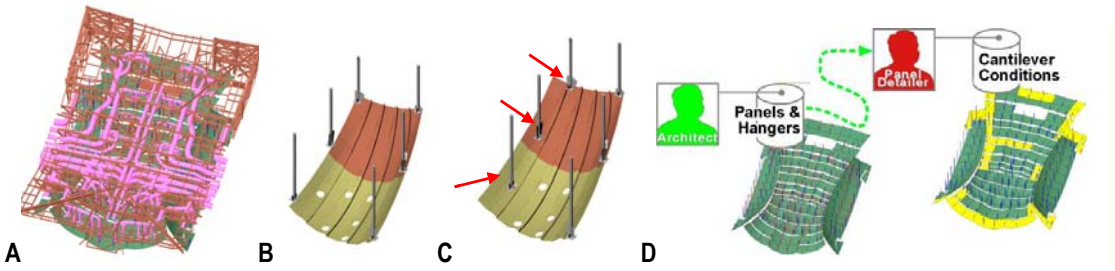
E. *The fabricator fabricated the beams. Using the Steel Framing view and the Deck Attachments view, he also welded each attachment onto the respective beam during the fabrication process in the controlled environment of the shop. The relationship between each beam and deck attachment is implicit in the view, so he needed to manually determine which attachment to weld to which beam. Other engineers, such as cost estimators and fabricators (not shown) also constructed and integrated dependent views of the Deck Attachments view.*

Figure 3: A portion of the workflow for the design and construction of the Walt Disney Concert Hall as it occurred on the project. The lines are dashed because the dependency between views is currently implicit.

As the project's architects and engineers collaborated and coordinated they iteratively constructed the Concrete Slabs and Steel Framing views, generating new metal decking attachment conditions, while eliminating others. The metal decking contractor needed to notice and annotate these new conditions in the Deck Attachments view. Missed conditions and slow propagation of design changes resulted in the inability to weld deck attachments in the shop. As a result, field welding of the deck attachments was required, and was much less efficient, accurate and safe than shop welding.

3.1.2 The cantilevered ceiling panels test case

Architects, engineers, contractors, subcontractors, and vendors collaboratively design the ceiling system (Figure 4A) of the WDCH (Post 2003). Ducts, catwalks, fire sprinklers, theater lighting, and several other systems vie for a tight space above 200 ceiling panels that measure approximately 3 x 4 meters (Figure 4B), weigh in excess of 1000 kg each, and hang from the roof trusses. Cantilever conditions occur where the edge of a panel extends significantly beyond the vertical steel tube hanger support (Figure 4C). The engineer responsible for framing the panels wants to keep track of the location, number, and severity of these conditions as he decides how to frame the panels. Generally, it is desirable to keep cantilever conditions to a minimum.



A. *The concert hall ceiling, from above with ductwork and with steel roof trusses overlaid.* ***B.*** *Two ceiling panels with no cantilever conditions.* ***C.*** *The same two panels after moving three supports causes four cantilever conditions.* ***D.*** *A mockup of a visualization that formal dependencies describing the cantilever conditions could help provide. The visualization highlights panels that have cantilever conditions.*

Figure 4: Images of the WDCH for the cantilevered ceiling panel test case.

The engineers on the WDCH project did not construct and maintain an explicit view of these cantilever conditions. Rather, these conditions were managed in an ad hoc fashion, based on the considerable engineering experience of the design team. This test case is therefore speculative: first, that no view was constructed because there are currently no tools to enable engineers to construct a new view by specifying its dependencies on other views, and second, that such a view could provide useful information for the panel detailer and for other engineers as they collaboratively design the ceiling system. Figure 4D shows a mockup view that highlights which panels contain cantilever conditions, to assist in duct routing.

3.2 Observation: Engineers could use a method to construct a geometric view from other views and control its integration

The test cases illustrate that AEC projects today are:

- Multi-disciplinary: The architect, structural engineer, steel detailer, deck detailer, steel fabricator, etc. all work for different organizations, representing different engineering criteria. They form a one-time, project-specific team to design, plan, and build a one-of-a-kind project in site-specific conditions. These engineers are contractually responsible for the information in their view(s), and therefore only they can specify the content of their perspectives. They cannot explicitly anticipate subsequent engineers' information needs.
- Constructive: To perform his task, the decking detailer constructs a geometric view of the deck attachments required on the WDCH. 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. Only the engineer responsible for a view can specify the dependencies on source views.
- 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 must become aware of modifications to source views through coordination meetings and amended documents[2], and must notice, and manually integrate their dependent geometric view.
- Unique: Deck attachments and cantilever conditions are an issue on some, but not all, AEC projects. The WDCH team did not formalize a priori how to construct a Deck Attachments view from Steel Framing and Concrete Slab views, or a Cantilever Conditions view from a Panels and Hangers view. Design concepts and approaches emerge within and across projects, and therefore engineers often need to construct new kinds of dependent views of changing source views.
- Time-consuming, Error-prone and Difficult: constructing and integrating the Deck Attachments view took the decking detailer over 140 hours, and over $160,000 worth of field welding was required. No cantilever view was constructed because it was too difficult to do so.

To design AEC projects today, engineers construct and integrate task-specific project views. These engineers construct these views from information in other engineers' views, and it is often important to keep these two views integrated. The transformation of the information in source views into information in the dependent view can be quite complex. As practiced today, this process of constructing and integrating views is often time-consuming, error-prone and difficult. Existing AEC project model approaches generally formalize a central model from which they derive task-specific views, or a federation of predefined and interrelated task specific views. Computer programmers are currently required to construct these dependencies. However, these approaches do not always adequately fit the characteristics of AEC projects where there is no single owner of a project model, and it is difficult to anticipate the information needs of the various engineers.

To address the difficulties observed in these test cases, we set the goal to enable engineers to more easily construct a task-specific view from information in other task-specific views and control the integration of this view as its source views are modified. A project model emerges from the iterative application of this method.

In order to enable engineers to work in this way, they could use a generic view with which they can construct their task-specific views of the project, and relate these views to other views. Many types of views could be constructed and integrated in a graph of dependencies. This research formalizes geometric views, because, as illustrated on the test

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

cases, geometry is a primary language of coordination and communication on AEC projects. The views used in AEC practice today contain a collection of geometric features. The geometric data types in these features consist of surfaces, lines, and points. The spatial interaction of these feature's data types causes engineering conditions to emerge between some but not all of these features. 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. While the features used in the WDCH did not have explicitly formalized relationships to features in other views, engineers could use a formal way to explicitly relate these features. For example, it would have been useful on the test case to formally relate each deck attachment to the beam and slab features that it attaches to, in order to facilitate the fabrication process. Automating the construction of the deck attachment view would enable engineers to rapidly consider the impact of changes in the slab or the beam geometry.

3.3 Requirements: For a project model approach that enables engineers to construct and control a project model of geometric views and dependencies between views

Based on these observations, Haymaker et al (2003a) formalized requirements for an approach that matches the characteristics of AEC projects, to enable engineers to easily construct and control a project model of views and dependencies. The following reviews our requirements for such an approach, focusing these requirements on the need to construct and integrate task-specific geometric views.

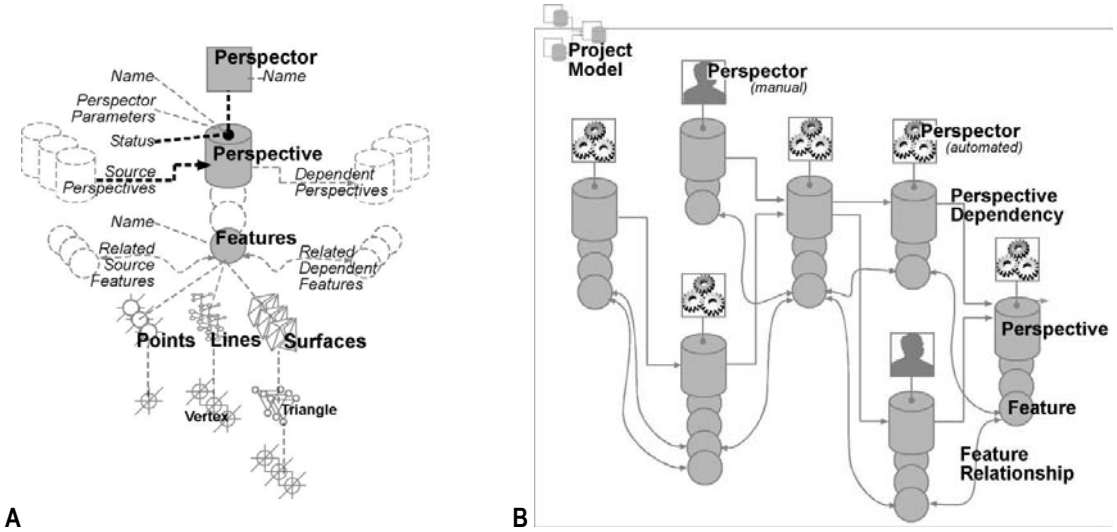
- **Generic Representation:**
    1. A conceptually simple, yet adequately expressive, generic engineering view containing geometric features with relationships to other features; for example, a generic view that engineers can use to describe beams, slabs, or deck attachments.
    2. A representation of the existence, nature, and status of the dependency of a view on other views; for example, tools to represent the dependency of the Deck Attachments view on the Concrete Slabs and Steel Framing views.
- **Generic Reasoning:**
    3. An intuitive way for engineers to specify to the computer how to construct a dependent geometric view from source geometric views; for example, enable the engineer to specify reasoning that constructs a Deck Attachments view from the Concrete Slabs and Steel Framing views.
- **Generic Management:**
    4. Methods for engineers to intuitively and 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 and integrate views from information in the Deck Attachments view.
    5. Methods for engineers to control the iterative construction and integration of this graph of views and their dependencies; for example, tools to enable an engineer to control the status of the Deck Attachments view with respect to the Concrete Slabs and Steel Framing views.

To address our requirements, this paper defines the Perspective Approach that is designed to enable engineers to easily, yet formally construct geometric views by specifying their dependency on other views, and control the integration of these views as their source views are modified. Specifically, in this paper we formalize this approach to construct and control the integration of geometric views.

4 The Perspective Approach: Enabling Engineers to Easily Construct and Control Views and Dependencies

The Perspective Approach enables engineers to iteratively construct views, called *Perspectives* from other Perspectives, and control the integration of these Perspectives as the project progresses. Conceivably there could be many different types of Perspectives, just as there are many types of project views such as schedules and cost estimates on AEC projects today. This paper formalizes *geometric Perspectives* (Figure 5A), consisting of Features that contain geometric data and relationships to other Features.

A geometric Perspective is similar to a CAD layer used in AEC practice today, except that it allows engineers to represent the existence, status and nature of its dependency on other geometric Perspectives. An engineer formalizes the nature of the dependency using a reasoning mechanism, called a *Perspector*. The formalization is modular, enabling an engineer to *compose* Perspectives and Perspectors into a graph, called a *Perspector Graph*, to perform sequences of transformations of information in source Perspectives into a dependent Perspective (Figure 5B). This modularity also enables engineers to *subsume* Perspector Graphs into one Perspector that will construct information in one Perspective from other Perspectives. A Project Model emerges as engineers iteratively compose and subsume Perspectors to construct integrated, task-specific Perspectives of other Perspectives. This section first describes how engineers formalize the dependency between Perspectives. The section then discusses the geometric Features implemented in this research to enable engineers to describe task specific geometric Perspectives. The section concludes by formalizing simple Management Processes that assist engineers in constructing and controlling task-specific Perspectives of other Perspectives as the project model evolves.



***A**. A geometric Perspective formalizes its dependency on other Perspectives. This formalization contains three parts (shown with darker dotted lines): the existence (Source Perspectives), the nature (Perspectors), and status of the dependency. **B.** Engineers can iteratively construct Perspectives from other Perspectives; an integrated project model emerges as a d.a.g. of Perspectives and dependencies. Perspectors can be manual (shown as a human icon) or automated (shown as a gear icon).*

Figure 5: Formalizing the dependency between task-specific views called Perspectives.

4.1 Perspectives: Engineering views that formalize their dependency on other Perspectives

A Perspective is a generic engineering view that formalizes its dependency on other Perspectives. Perspectives formalize their dependencies into three parts:

- **Existence**: Engineers represent the existence of the dependency of the Perspective on other Perspectives using an ordered list called the Source Perspectives relationship. When this relationship is established, the source Perspective represents the inverse of this dependency using the dependent Perspectives relationship, which is used by Management Processes to control integration (explained in Section 4.2).
- **Status***:* Engineers can see the Integration Status of this dependency. This Integration Status is set by Perspectors as they construct Features in Perspectives. A simple Integer (0 = integrated, 1 = not integrated, 2 = being integrated) is used to represent this status.
- **Nature**: Engineers use the Perspector relationship to specify the nature of the dependency of the Perspective on source Perspectives. As shown in Figure 5A, a Perspective can also contain Perspector Parameters that a Perspector can use to further specify how the Perspector should construct the information in the dependent Perspective. These Perspector Parameters enable a Perspector to be used more generally on many Perspectives (Perspectors are explained in greater detail in Section 4.1.1)

Figure 6 diagrams how engineers can use geometric Perspectives for the deck attachments test case: The steel detailer constructs a Steel Framing Perspective containing beam Features (Features are described in Section 4.1.2). The architect constructs a Concrete Slab Perspective containing concrete slab Features. The deck detailer constructs a Deck Attachments Perspective containing deck attachment Features by specifying the existence of the dependency on the Concrete Slabs Perspective and on the Steel Framing Perspective using the source Perspectives relationships (arrows) and specifying the nature of this dependency using the Find Deck Attachments Perspector. The status of the dependency is represented using the Perspective's status attribute, and can be controlled using the simple Management Processes (described in Section 4.2) as the design progresses. The figure shows the definition of an instance of a Deck Attachments Perspective, while only showing the name and a graphical representation of instances of Steel Framing and Concrete Slabs Perspectives.

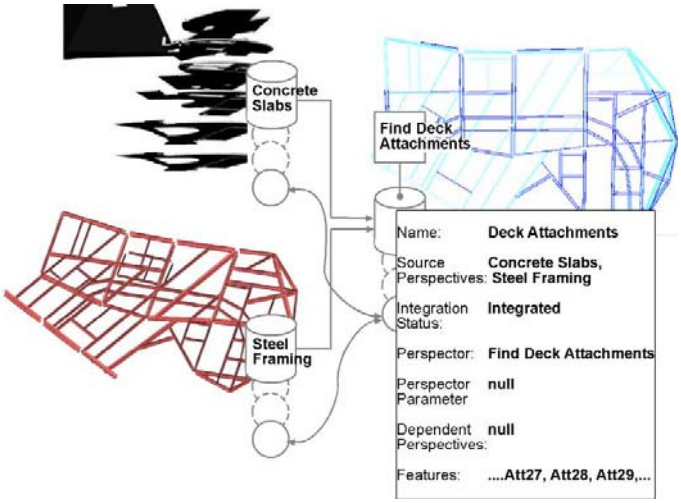


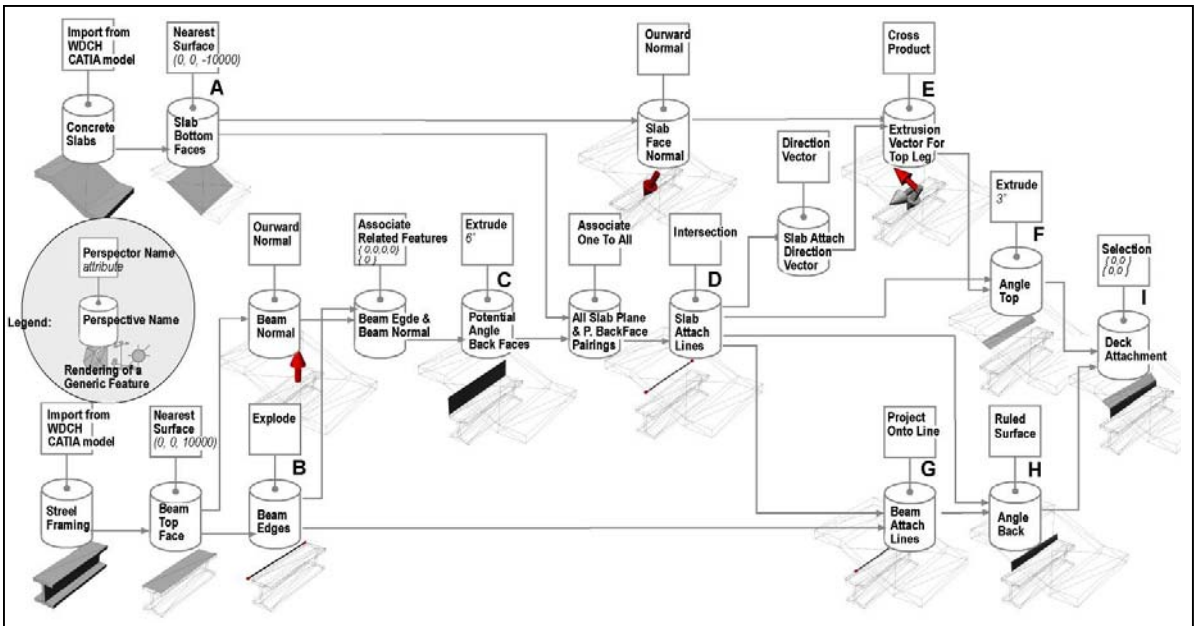
Figure 6: The Find Deck Attachments Perspector constructs deck attachment Features in the Deck Attachments Perspective by analyzing the concrete slab Features in the Concrete Slabs Perspective and the steel framing Features in the Steel Framing Perspective. The Deck Attachments Perspective definition is shown in full.

The existence of the dependency is represented with a simple ordered relationship, the status is a value, and the nature is formalized as a reasoning mechanism, called a Perspector, which constructs an engineer's dependent Perspective from source Perspectives. Requirement 3 states that engineers need an intuitive method to specify the reasoning that constructs a Perspective from other Perspectives. The method proposed in this research is to guide engineers to iteratively apply the method discussed above: construct Perspectives from other Perspectives using a Perspector. Haymaker et al (2003b) discuss Perspectors in detail. The next section summarizes how Perspectors work.

4.1.1 Perspectors: Composable, subsumable, modifiable reasoning to formalize the nature of the dependency of a Perspective on source Perspectives

A Perspector enables an engineer to formalize the nature of a Perspective's dependence on its source Perspectives. In other words, a Perspector contains a reasoning method to construct information in the dependent Perspective based on information in the source Perspectives. A Perspector can be automated, containing algorithms that automatically construct this information, or it can be manual, providing User Interface tools that allow the engineer to construct this information.

Because a Perspector's input and output are both Perspectives, Perspectors are composable and subsumable. Perspectors are composable in that Perspectives can be composed into graphs using their dependencies, and thus can formally define how to perform a series of transformations on source Perspectives to construct a dependent Perspective. Perspectors are subsumable in that a graph of Perspectors, and their associated Perspectives, can be subsumed into one Perspector. For example, Figure 7 shows a representative portion of a *Perspector Graph* that has been composed to automatically construct deck attachments in a Deck Attachments Perspective from beams and slabs in Steel Framing and Concrete Slabs Perspectives. The graph of Perspectors constructs a series of Perspectives that first analyzes where the deck attachments are needed, and then constructs the Deck Attachments Perspective. This graph can be subsumed into the Find Deck Attachments Perspector shown in Figure 7. In addition to being composable and subsumable, Perspectors are modifiable, because their relationships and attributes can be edited. A Perspective can contain Perspector Parameters that a Perspector uses when constructing a Perspective's Features. These parameters enable more generality for each Perspector. For example, the Extrude Perspector in Figure 7C accepts a parameter to specify how far to extrude the geometry in each Feature it is given. The same Extrude Perspector can be reused in other Perspectives that contain other values for this parameter. For example, Extrude is used again with a different parameter at Figure 7F. The Find Deck Attachments Perspector Graph is described in greater detail in Haymaker et al (2003b), where we provide evidence that it may be possible to exploit this modularity to define a relatively small language of predefined geometric Perspectors that engineers can modify, compose and subsume to describe many kinds of complex but useful dependent Perspectives of iteratively changing source Perspectives. The composable, subsumable, modifiable formalism of Perspectors and Perspectives make them potentially intuitive for engineers to use.



A. *Reformulate each slab Feature into bottom face Features.*
B. *Reformulate each beam Feature into individual beam top edge Features.*
C. *Generate potential angle back face Features by extruding each beam edge along the associated beam top face normal a distance of six inches.*
D. *Generate a slab attach line Feature representing where potential angle back faces intersect slab bottom faces. This represents where a deck attachment should connect to a slab.*
E. *Generate the cross product of this slab attach direction vector and the bottom face of the slab, to find the appropriate direction to extrude the slab attach line, so that it extrudes away from the interior of the beam's face.*
F. *Generate each angle top by extruding each slab attach line along extrusion vector 3 inches.*
G. *Generate each beam attach line by projecting each slab attach line onto its corresponding beam edge.*
H. *Generate each angle back by ruling a surface between each slab attach line and corresponding beam attach line.*
I. *Generate each deck attachment by grouping each angle back with the corresponding angle top.*

Figure 7: The Deck Attachments Perspector Graph: An engineer (in this case the lead author of this paper) composes and modifies domain-independent Perspectors to automatically construct a Deck Attachments Perspective from Concrete Slabs and Steel Framing Perspectives.

Perspectors are generic reasoning mechanisms that engineers can use to establish the nature of the dependency between Perspectives. That is, they construct Features in the dependent Perspective based on Features in the source Perspective. The next section discusses the geometric Features implemented in this research to enable engineers to describe task-specific concepts with which to construct task-specific geometric Perspectives like those described in the test cases. First Features are defined, and then Features are applied to the deck attachments test case.

4.1.2 Features: geometric concepts with which to construct a geometric Perspective

In order to enable engineers to describe geometric Perspectives, this research uses a definition of Feature that is congruent with that of Dixon's research. A Feature contains geometric data to describe a task-specific engineering concept (Dixon and Poli1995). To construct a task-specific geometric Feature, a Perspector or an engineer in the context of a Perspector gives the Feature a name, constructs geometric data consisting of Surfaces, Lines, and/or Points, and relates the Feature to any Related Source Features (potentially any Feature considered while constructing this Feature, or only specific Features according to algorithm encoded in the Perspector). When this relationship is established, the Perspector can also represent the inverse of this relationship from the source Feature to the

dependent Feature in the Related Dependent Feature relationship. The Perspector aggregates these Features into a Perspective to describe a task-specific project view that is very similar to a CAD layer today, except the Features can contain explicit relationships to other Features.

Figure 8 diagrams how Perspectors (or engineers in the context of Perspectors) could use a generic geometric Feature to construct task-specific Features for the deck attachments test case: The steel detailer constructs a steel beam Feature using a geometric Surface to describe a beam's boundary. The architect constructs a slab Feature using a geometric Surface to describe a slab's boundary. The deck detailer constructs a deck attachment Feature using a single Line along the edge of the beam where the attachment is required, as was done in the test case, or (using the Find Deck Attachment Perspector) he constructs the deck attachment Feature using a Surface that describes the back face of the deck attachment to represent not just the length and position, but also the size of the deck attachment, as is done in Figure 7. The deck detailer also constructs a relationship between the deck attachment Feature and the beam and slab Features that it connects. The figure shows the full definition of an instance of a deck attachment Feature, while only showing the name and a graphical representation of the slab and beam Features. This deck attachment Feature has a Name, Related Source Features (the slab and beam that it connects), Related Dependant Features (currently none), one Surface, no Lines, and no Points. Perspectives and Features are also used to describe the various parts of these building components, and several spatial relationships between these components as the Perspector Graph in figure 7 iteratively constructs deck attachments from slabs and beams.

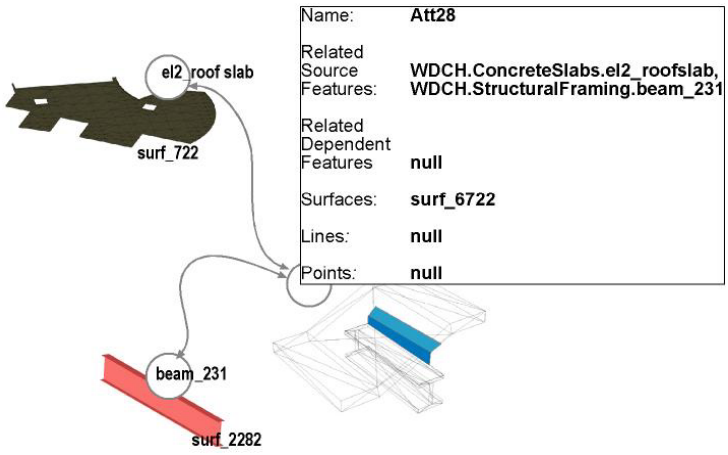


Figure 8: Beam, slab, and deck attachment Features: The figure shows the definition of a deck attachment Feature completely. These Features are aggregated in the Structural Framing, Concrete Slab, and Deck Attachment Perspectives.

4.1.3 Comparing the Perspective Approach to our requirements

In summary, Perspectives are task-specific engineering views that engineers can use to formalize their dependency on other Perspectives. Together with the Features, Surfaces, Lines and Points implemented in this paper, Perspectives satisfy requirements 1 and 2 as defined in section 3.3. Geometric Perspectives are generic, they are used to describe views of slabs, beams, deck attachments, many of the parts of these building components, the spatial relationships between these components, and the dependencies between these views. Throughout the example, the engineers

need only focus on the Perspectives and the dependencies between Perspectives. Other data types, such as Solid Models and NURBS, or other ways to represent a view than a collection of Features can be used without modifying the contributions in this research, and future work may incorporate these data types and other types of views. However, engineers are neither computer programmers nor geometry experts; understanding the trade-off between increased expressiveness versus the increased learning curve for engineers will need to be considered. Perspectors are modular reasoning mechanisms that engineers can modify, compose, and subsume to construct a dependent Perspective from source Perspectives. In this way they satisfy requirement 3.

The next section shows that the d.a.g structure of Perspectors and Perspectives can be exploited to formalize intuitive Management Processes that satisfy requirement 4 and 5.

4.2 Management Processes to control a project model of Perspectives and their dependencies

The previous section describes how engineers can construct a Perspective by formalizing its dependency on other Perspectives. This section describes Management Processes that are designed to assist engineers in controlling the integration status of their Perspectives with respect to the Perspectives on which they depend.

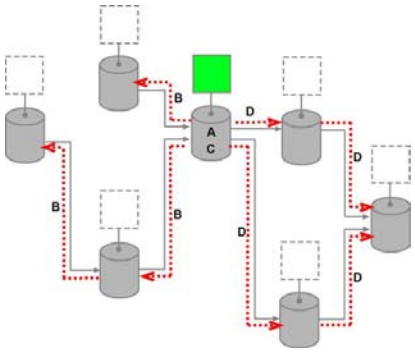
The first Management Process simply assures that the dependencies between Perspectives are properly constructed:

> ***Management Process 1:*** *When constructing a new dependent Perspective, construct a reference to the source Perspective in the dependent Perspective's source perspective list, and place a reference to the dependent Perspective in each source Perspective's dependent Perspective list.*

The second Management Process simply assures that the integration status of all Perspectives is up to date with respect to the iteratively modified source Perspectives on which they depend:

> ***Management Process 2:*** *Before (re)constructing a Perspective, check that each source Perspective's Integration Status is set to Integrated. While (re)constructing a Perspective, set that Perspective's Integration Status to Being_Integrated. After (re)constructing a Perspective, recursively set all dependent Perspectives' Integration Status to Not_Integrated.*

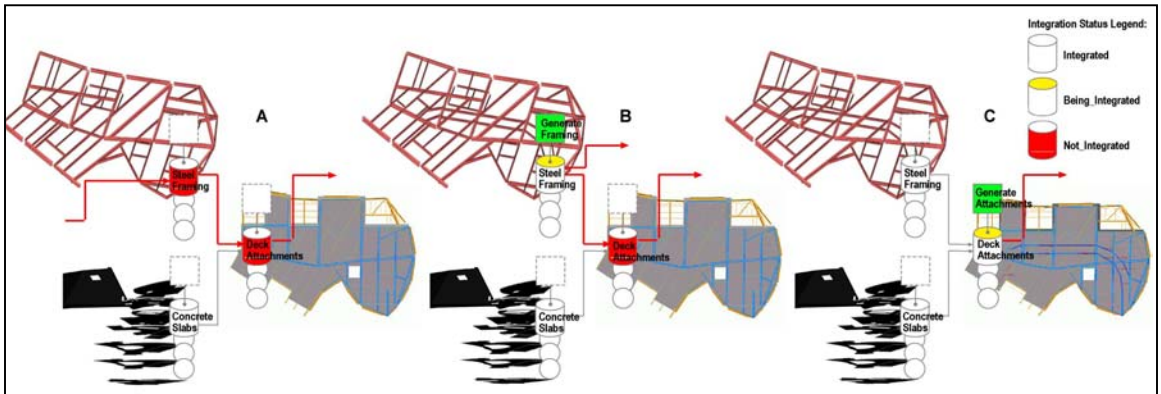
Figure 9 formalizes this Management Process in diagram form, showing that a Perspector implements this process.



A. *The Perspector designates its Perspective as Being_Integrated and* ***B.*** *checks that all source Perspectives are designated as Integrated. If any is not, call on the source Perspective's Perspector to integrate that source Perspective (this results in recursion upstream through all source Perspectives, assuring they are integrated).* ***C.*** *The Perspector then constructs the Features in its Perspective, designates that Perspective as Integrated, and,* ***D.*** *designates all dependent Perspectives (recursively) as Not_Integrated.*

Figure 9: Management Process for controlling Integration. When constructing Features in a Perspective using a Perspector (the solid square).

Figure 10 illustrates Management Process 2, using the deck attachments test case.



A. *A Perspector constructs a source Perspective (not shown) of the Steel Framing Perspective and sets the Integration Status of all dependent Perspectives recursively down the graph, including the Steel Framing Perspective, to Not_Integrated to inform other interested engineers and Perspectors that information in one of their source Perspectives has changed.*

B. *The steel detailer is notified in a graph Viewer that the Steel Framing Perspective's status is Not_Integrated. He invokes the Generate Framing Perspector that encodes the nature of the dependency of the Steel Framing Perspective on its source Perspective(s). The Generate Framing Perspector sets the Steel Framing Perspective's Integration Status to Being_Integrated and all dependent Perspective's Integration Status to Not_Integrated. When the Generate Framing Perspector finishes constructing the Features in the Steel Framing Perspective, it sets the Steel Framing Perspective's Integration Status to Integrated.*

C. *The metal decking detailer notices that the Deck Attachments Perspective's status is no longer Integrated. He invokes the Find Deck Attachments Perspector that he composed to automatically construct the Deck Attachments Perspective from the Concrete Slabs and Steel Framing Perspectives. After checking that each source Perspective's status is Integrated, the Find Deck Attachments Perspector sets the Deck Attachments Perspective's status to Being_Integrated, constructs the deck attachment Features, sets the Deck Attachments Perspective's status to Integrated, and finally sets the Integration Status of all Perspectives that are dependent on the Deck Attachments Perspective to Not_Integrated.*

Figure 10: Diagram of the behavior of the Management Process for constructing and controlling the integration of Features, based on the test case.

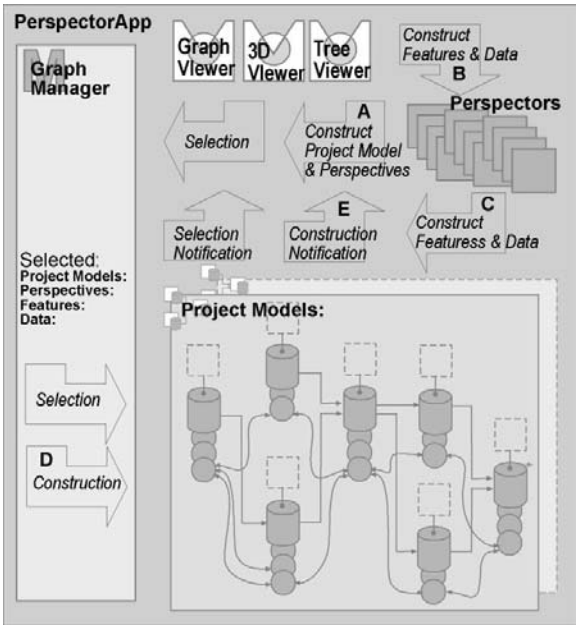
The two simple Management Processes presented in this paper are designed to enable a multidisciplinary team of engineers to construct and control Perspectives and dependencies to generate an emerging Project Model. The next section describes the implementation of Perspectives and the Management Processes into a prototype that enables an engineer to construct and control multiple Perspectives and their dependencies. In subsequent sections, analysis of using this prototype provides evidence for the power and generality of the Perspective Approach.

5 PerspectorApp: A prototype of the Perspective Approach

This section describes the implementation of the Perspective Approach into a working prototype, called PerspectorApp. This research implemented PerspectorApp on a single computer in order to investigate the feasibility of the Perspective Approach. PerspectorApp enables an engineer to construct Perspectives by formalizing their dependencies on other Perspectives, and control the integration of the Perspectives as they are iteratively modified. A Project Model emerges, consisting of Perspectives (and their dependencies, including Perspectors), Features (and their relationships), and geometric data in Features. In PerspectorApp a Project Model is simply a collection of Perspectives. The Project Model abstraction is useful to engineers working with PerspectorApp, it is not used in managing the dependencies between Perspectives, and therefore a Perspective can conceptually exist in several Project Models.

Figure 11 shows two mechanisms that were used to implement the Perspective Approach into a working prototype:

- **Viewers** maintain visualizations that enable the engineer to interact with and construct Project Models, Perspectives, Features, and data. Different Viewers interact with Project Models in different ways.
- **A Graph Manager** assures that Viewers have updated visualizations of the Project Models, Perspectives, Features and data. The Graph Manager intercepts Viewer interactions and performs all construction of Project Models, Perspectives, Features and data. When finished, the Graph Manager notifies registered Viewers of this construction.



***A**. Viewers implement Graph Manager interfaces to construct a Project Model, or Perspective. **B**. Engineers use Perspectors that implement Graph Manager interfaces **(C)** to construct Features and Data. **D**. The Graph Manager reacts to the requests from Viewers or from Perspectors by making the appropriate construction and the Graph Manager notifies **(E)** the Viewers that have registered for notification of construction of Data, Features, Perspectives, and/or Project Models.*

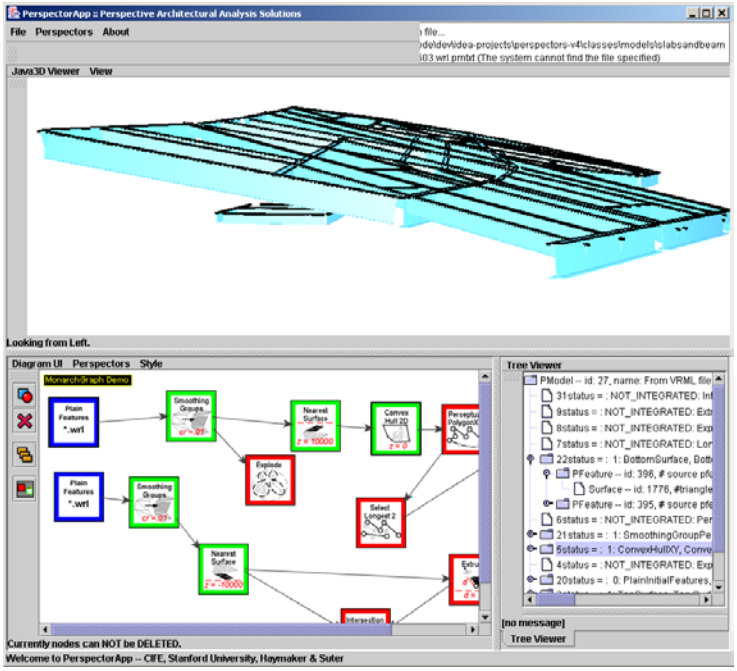
Figure 11: A diagram of PerspectorApp. Engineers select and construct Project Models, Perspectives, Features, and Data through Viewers. These Viewers use interfaces (shown as large, hollow arrows) provided by the Graph Manager.

5.1 Viewers: Enabling engineers to see, select, and construct the model

Engineers use Viewers to visualize, select, and construct (i.e., add, modify, or delete) Project Models and their parts (i.e., Perspectives (and dependencies), Features (and relationships), and data). When a Viewer is added to PerspectorApp, it registers with the Graph Manager (described below) to be notified when a Project Model or its parts are constructed. Viewers implement *construction interfaces* provided by the Graph Manager: The interfaces are Graph Manager methods that each Viewer calls in order to construct part of a Project Model. When a Viewer calls one of these methods, the Graph Manager constructs the information, then notifies all Viewers that have registered for notification of a construction. In this way, a Viewer can maintain an up to date visualization of the Project Models and their parts.

Different Viewers are useful for interacting with the Project Models and their parts in different ways. PerspectorApp implements three Viewers, shown in Figure 12: The Java 3D Viewer enables engineers to see a 3D visualization of the Features in Perspectives, and to select and construct Features and geometric data contained in Features. The Diagram Viewer enables the engineer to view the graph of Perspectives, construct new Perspectives, and control the integration status of Perspectives by invoking the Perspector of a Perspective that is Not_Integrated. The Tree Viewer enables the engineer to interact with the Project Model (Perspectives, Features, Data) hierarchically. An engineer can

click on a Project Model to visualize its Perspectives, click on a Perspective to visualize its dependencies and Features, and open Features to visualize its geometric data and relationships.



A Java3D Viewer (top) provides an interactive 3D visualization of selected Perspective's Features. The Graph Viewer (bottom left) provides an interactive view of Perspectors and their dependencies (because every Perspective has one associated Perspector, the dependencies can be drawn between Perspectives, as shown in previous Figures, or between Perspectors, as shown here). The Tree Viewer (bottom right) provides an interactive hierarchical view of a Project Model. Project Models decompose into Perspectives, which decompose into Features, which decompose into data.

Figure 12: The three Viewers implemented in PerspectorApp.

5.2 Graph Manager: Keeping viewers up to date

The Graph Manager contains interface methods to add, modify, and remove Project Models, Perspectives (and dependencies), Features, and data. Viewers call these methods when a user interaction requests construction. The Graph Manager contains a reciprocal set of methods to notify all Viewers that a Project Model, Perspective, Feature or data has been constructed, and maintains a list of all Viewers that request notification of the construction. The Graph Manager also implements a similar set of methods to select Project Models, Perspectives, Features and data, and to notify registered Viewers of the selection. Different Viewers choose to represent selection in different ways, usually by visually highlighting the selected items.

It is important to point out that PerspectorApp handles the integration between a Viewer and a Project Model (and its parts), and integration between Perspectives differently. The Graph Manager notifies registered Viewers instantaneously of any construction of a Project Model, Perspective, Feature, or data. Viewers are programmed to automatically update their visualizations. PerspectorApp gives engineers more explicit control over the existence, status, and nature of the dependencies between Perspectives.

6 Results: Formal, Integrated Task-Specific Perspectives on the WDCH

This section discusses the results of applying Perspector App to the test cases from the WDCH. These results provide evidence for a discussion of power and generality of the Perspective Approach.

The research applied the Perspective Approach to the deck attachments test case. We exported a collection of ninety-seven steel beam features and two concrete slab features that comprise the roof of a portion of the WDCH called Element 2 from the WDCH CATIA model to a VRML[3] file using CATIA VRML export functions. This collection represents approximately 0.5 percent of the total steel and concrete on the job. All of the deck attachments in this area required field welding because the existing manual process did not construct a deck attachment view of the conditions in time to enable shop welding. We then implemented a VRML import Perspector and imported the VRML files into PerspectorApp: We converted each VRML concrete slab Group[4] into a Feature and imported these Features into a Concrete Slabs Perspective, and we converted each VRML steel beam Group into a Feature and imported these Features into a Steel Framing Perspective. Figure 13A shows a 3-D view of these two Perspectives. We then implemented the collection of Perspectors and composed the Perspector Graph described in Figure 7 to automatically construct Features in a Deck Attachments Perspective from Features in Concrete Slabs and Steel Framing Perspectives.

Figure 13B shows PerspectorApp results: a Perspective of the required Deck Attachments for the Element 2 roof steel. We used generic Perspectives and Features to describe task-specific views of concrete slabs, steel beams, various parts of these building components, and several spatial relationships between these parts as the Perspector Graph iteratively constructs deck attachments from slabs and beams. We subsumed this Perspector Graph into the Find Deck Attachments Perspector that constructs the required Deck Attachment Features in the Deck Attachments Perspective from the Concrete Slabs and Steel Framing Perspectives.

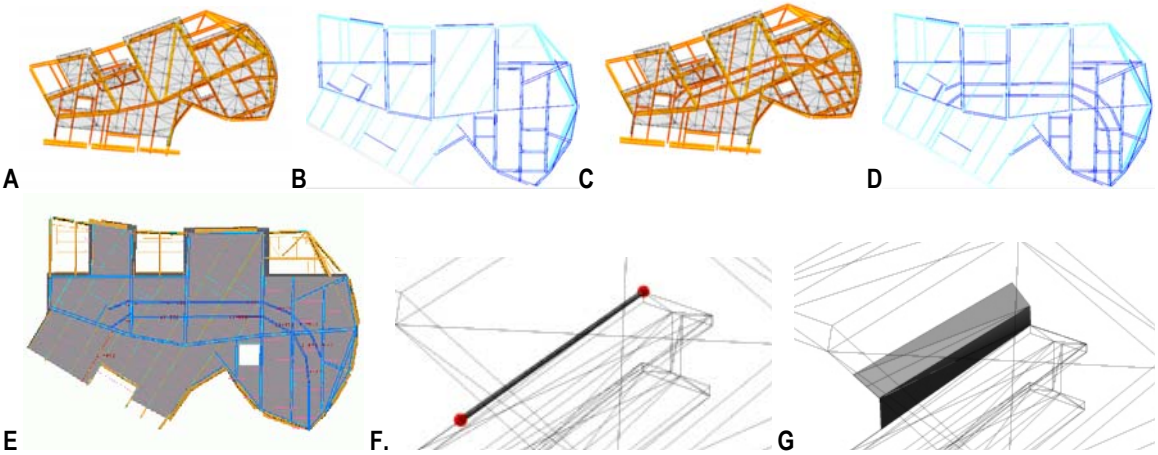
Figure 13C shows the same roof after a design change (the addition of several beams to support additional roof loading of a window washing crane). When the design change was imported from the WDCH CATIA model using the VRML import Perspector, the Management Processes automatically changed the Deck Attachments Perspective's Integration Status to Not_Integrated. After re-running the Find Deck Attachments Perspector, Figure 13D shows the PerspectorApp results on the Element 2 roof inclusive of this design change, identifying 114 Deck Attachment conditions. The Management Processes automatically changed the Deck Attachments Perspective to Integrated. Figure 13E shows an as-built model, constructed by the WDCH engineers, showing the deck attachments that were actually installed on the Element 2 roof.

---

[3] VRML stands for Virtual Reality Modeling Language – a relatively simple, object oriented data format that can be used to describe task-specific geometric views.

[4] Group is a VRML object that can be used to aggregate Geometry, or other objects. Groups are used much like features, although Groups do not explicitly contain relationships to other Groups.

Figure 13F shows the amount of detail the WDCH engineers used when manually constructing the deck attachments. The WDCH deck attachment is modeled as a line, from which a fabricating engineer can determine the length and location of each deck attachment; however, further work was required to determine the complete dimensions of the deck attachment. Figure 13G shows the amount of detail constructed using automated Perspectors in the Perspective Approach. The Find Deck Attachments Perspector models two surfaces of the deck attachments. This added detail would enable a fabricating engineer to determine the length, width and precise location of each deck attachment.

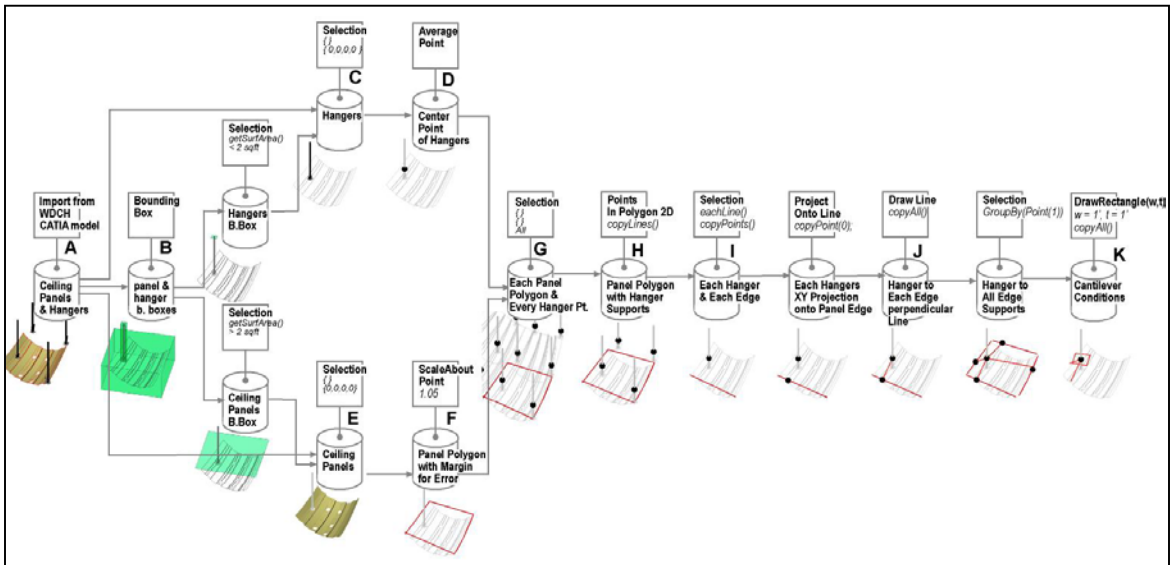


A. *Steel Framing Perspective and Concrete Slab Perspective for Element 2 Roof.* ***B****. PerspectorApp results on Element 2 roof before design change.* ***C.*** *Element 2 roof after design change; steel framing is added to support a window-washing crane.* ***D.*** *PerspectorApp results on Element 2 after the design change.* ***E.*** *As-built conditions for Element 2 roof.* ***F.*** *The deck attachment as constructed on the WDCH: A single line along each beam where a deck attachment is required.* ***G.*** *The deck attachment as modeled using Perspectors: A surface describing the back faces of the deck attachment angle.*

Figure 13: Images of the validation of the Perspective Approach on the deck attachments test case.

The research also applied the Perspective Approach to the Cantilevered Ceiling Panel test case. We exported a collection of fifty-three ceiling panels and forty-eight hangers that comprised approximately 20% of the WDCH Ceiling from the WDCH CATIA model using CATIA's VRML export functionality. In this case we exported both panels and hangers into the same VRML file, losing the explicit distinction between ceiling panels and hangers. Rather than go back to the CATIA model and re-export into two separate files, we chose to use Perspectors to regenerate this information automatically.

The research reused several Perspectors from the Find Deck Attachments Perspector Graph, implemented several more, and composed the Find Cantilever Perspector Graph to automatically construct a Perspective that finds and formally represents cantilever conditions. The graph shown and described in Figure 14 omits some detail; the Cantilever Conditions Perspector Graph is described in detail in Haymaker et al (2003b).



A. *The original Perspective contains undifferentiated ceiling panel and hanger Features, with no explicit information distinguishing between the two.*

B. *Generate a bounding box around each Feature.*

C. *Select the bounding box Features with top face less than 2 square feet, select the original associated Feature (at A), copy the geometry, and call this Feature a hanger.*

D. *Reformulate each hanger into the center point of the hanger.*

E. *Similar to C. If the area of the top face of the bounding box is greater than 2 square feet, select the original associated Feature (at A), copy the geometry, and call this Feature a panel.*

F. *Reformulate this panel into an approximate representation of the edges of each panel using a convex hull algorithm that generates a polygon around these surfaces.*

G. *Select each panel edge, and every hanger center point.*

H. *Select the points that are contained within the panel edges. This represents which hangers support each panel. An individual hanger can support more than one panel.*

I. *Select each panel edge and each hanger center point.*

J. *Project the hanger center point onto the panel edge, and generate a line from the center point to this projection point. This line represents the distance from the center point to each respective edge.*

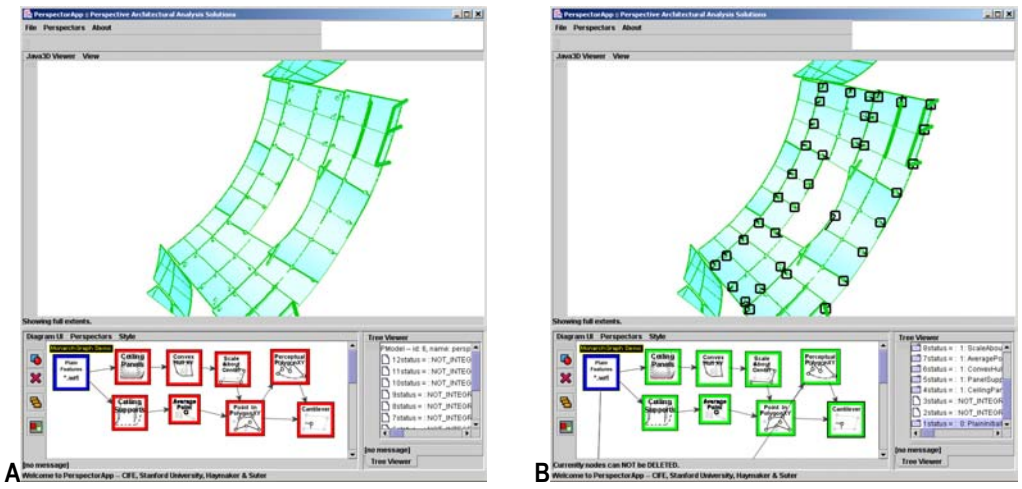
K. *Select the 4 perpendicular distance lines corresponding with each hanger center point; select the two shortest of these; select the lines that are longer than the cantilever distance. Draw a square around the corresponding point in the Feature. These are the cantilever conditions.*

Figure 14: The Cantilever Conditions Perspector Graph: Engineers can modify, compose and subsume low-level geometric Perspectors to automatically construct a Perspective that describes cantilever conditions between ceiling panels and their supporting hangers.

Geometric Perspectives are general. We used Perspectives and Features to describe task-specific views of ceiling panels and hangers, various parts of these building components, and several spatial relationships between these parts as the Perspector Graph iteratively constructs a view of the cantilever conditions between these panels and hangers. We subsumed this Perspector Graph into the Find Cantilever Conditions Perspector that constructs the required cantilever condition Features in the Cantilever Conditions Perspective from the Ceiling Panel and Hanger

Perspectives. Throughout the example, the engineers need only focus on the Perspectives and the dependencies between Perspectives.

Because the dependencies between these task-specific Perspectives are formalized, an engineer (in this case, the lead author of this paper) can use the Management Processes to construct and control the cantilever conditions as another engineer (in this case, also the lead author) iteratively adds and moves panels and panel hangers. Figure 15 shows the implementation of the Cantilever Ceiling Panel Perspector Graph in Perspector App.



A. *The Ceiling Panels and Hanger Perspectives,* ***B.*** *A Cantilever Conditions Perspective is formally constructed by running the Find Cantilever Conditions Perspector. Cantilever Features associate a ceiling panel with the hanger Feature that both supports it, and creates a cantilever Feature. A cantilever Feature contains a closed Line that draws a square around the support, and relationships to the panel and hanger.*

Figure 15: Implementation the Perspective Approach on an industrial test case: the Cantilevered Ceiling Panel Perspector Graph in PerspectorApp.

7 Discussion: Better, Faster, Cheaper Integrated Project Models for AEC

Using test cases from the WDCH, this research shows that engineers may benefit from a project modeling approach that enables them to easily construct and control task-specific views and their dependencies. Previous sections: (1) formalized geometric Perspectives (and their dependencies) that contain Features (and their relationships), and data in Features, and (2) formalized simple Management Processes that enable engineers to iteratively construct and control Perspectives form other Perspectives, thus generating an evolving, integrated Project Model. A prototype called PerspectorApp shows that engineers can use the Perspective Approach to formally plan, design, and execute their project to mitigate the difficulties they encounter on industrial scale projects such as the WDCH.

The power of the Perspective approach is demonstrated by the deck attachment test case. Geometric Perspectives are used to represent beams, slabs, deck attachments, aspects of these components, and spatial interactions of these components. The representation also adequately describes the dependencies between these Perspectives for the test cases. An engineer composed Perspectives and Perspectors into a Perspector Graph, which automatically constructs

a Deck Attachments Perspective more quickly and accurately and with more useful detail than current, manual practice allows. Once composed, the Find Deck Attachments Perspector becomes a formal procedural specification for the deck attachment conditions, which can be reused on subsequent projects. The Management Processes enabled the engineer to construct and control an integrated Perspective (the Deck Attachments Perspective) from the iteratively modified Perspectives of other engineers (the Concrete Slabs and Steel Framing Perspectives).

The generality of the Perspective Approach is demonstrated by the cantilever conditions test case. The same generic representation and Management Processes apply to two very different kinds of problems: automating the design of deck attachments between slabs and beams, and automating the analysis of an architectural ceiling system for cantilever conditions between architectural ceiling panels and the hangers. The representation adequately describes views of panel hangers, the center point of the panel hangers, ceiling panels, the boundaries of ceiling panels, an association that describes which hangers support which panels, and finally, which of these support conditions constitutes a cantilever condition. The representation also adequately describes the existence, status and nature of dependencies between these Perspectives, and the Management Processes enables engineers to easily control the integration of these Perspectives.

The limitations of the Perspective Approach formalism presented in this paper include: (1) Limited Representation: Any representation is an abstraction. The data types implemented in Features are Surfaces, Lines, and Points. Other geometric data types, such as NURBS, Solid Models, other non-geometric data types, and more complex view structures than a collection of Features can increase the expressive power of Perspectives, however potentially at the expense of greater complexity for engineers who need to understand their Perspectives and the Perspectors that transform them. (2) Limited Reasoning: While they perform well on the test cases, the individual Perspectors and Perspector Graphs presented in this paper are only provisionally tested. They are an initial investigation into the power and generality of the Perspective Approach. Engineers could construct deck attachments or cantilever conditions Perspectives in different ways, thus composing varying, progressively better, Perspector Graphs. (3) Limited Management: The Perspective approach does not support cycles in dependencies due to the acyclic nature of the formalism. While acknowledging that the dependencies between views can often be cyclical--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 (d.a.g.) of views and their dependencies to address the multi-disciplinary, constructive, iterative, and unique nature of AEC projects. (4) Limited Implementation: PerspectorApp is currently formalized to run on a single computer, with a single engineer, issues of remote location of Perspectors and Perspectives across a network, version management, access control, computational performance and the building of UI tools to enable manual Perspectors were scoped out of the research.

The future work for the Perspective Approach will focus on understanding the appropriate Perspective representations for specific tasks, techniques for remotely locating, organizing and managing Perspectives and Perspectors across a

network, and developing a language of Perspectors. By applying the Perspective Approach to more AEC design automation and analysis test cases, a set of future research questions is expected to emerge: (1) How general is the formalization of a geometric Perspective? In other words, in what contexts are new data types such as NURBS, Solid Models, non-geometric data types, and views of a different structure required? (2) How can Perspectors be remotely located over a network to enable engineers to construct task-specific Perspectives from other engineers' remotely located Perspectives? (3) How can cycles be incorporated to enable design optimization?

To date, it has been difficult to formalize integrated models for AEC projects because of their multi-disciplinary, constructive, iterative, and unique nature. Many researchers recognized the need for model evolution. This paper formalized the Perspective Approach, which is designed to enable an integrated project model to evolve from the iterative interaction of engineers with information from multiple disciplines. The paper provides evidence, using industrial scale test cases, that the Perspective Approach may empower engineers from multiple disciplines to coordinate and integrate their task-specific views, engage in automated design and analysis, and facilitate more efficient design iteration and project execution. These early empirical results suggest that the Perspective Approach may prove practical in practice.

8 References

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

Autodesk (2003). 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, August 2003.

Bentley (2003). Microstation Triforma www.microstation.com.

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

Clayton, M., Teicholz, P., Fischer, M., and Kunz J.(1999). “Virtual Components Consisting of Form, Function, and Behavior.” Automation in Construction, 8, 351-67.

Dassault Inc. (2003). CATIA R7, www.catia.com.

Dixon J., and Poli, C. (1995). “Engineering Design and Design for Manufacturing.” Field Stone Publishers.

Eastman, C. and Jeng, T-S. (1999). “A Database Supporting Evolutionary Product Model Development for Design.” Automation in Construction, 8 (3), 305-33.

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

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., Fischer, M., Kunz, J., and Suter, B. (2003a). “Engineering Test Cases to Motivate the Formalization of a Project Model as a Directed Acyclic Graph of Geometric Views and Their Dependencies,” Working Paper Nr 080, CIFE, 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 081, CIFE, 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 Operability* http://cic.vtt.fi/niai/technical/IFC_2X/index.htm.

Khedro, T. and 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, eds. John Gero and Fay Sudweeks, 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.

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.

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.

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).

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

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

Sriram D. Ram, *Distributed & Integrated Collaborative Engineering Design*, Sarven Publishers, 2002.

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

Tekla (2003) *XSteel*, www.xsteel.xom.

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

Van Nederveen, G. A., and Tolman, F. P. (1992). “Modelling Multiple Views on Buildings.” *Automation in Construction*, Vol. 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.