CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

# PERSPECTORS: Automating the Construction and Coordination of Multidisciplinary 3D Design Representations

By

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

CIFE Technical Report #145
April 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*

# PERSPECTORS: Automating the Construction and Coordination of Multidisciplinary 3D Design Representations[1]

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

Stanford University, Department of Civil and Environmental Engineering
Center for Integrated Facility Engineering (CIFE), Bldg. 550, Rm. 553H,
Stanford, CA, USA 94305
haymaker@stanford.edu, bsuter@stanford.edu, kunz@stanford.edu, fischer@stanford.edu

Keywords: 3D, 4D, Coordination, Design, Process models, Product models, Virtual Design and Construction

[1] Presented at the AAAI Spring Symposium 2003

**Abstract**

We formalize a multidisciplinary project model as a directed acyclic graph of dependencies between representations. For the nodes of this graph, we formalize a generic representation, called a "perspective," which contains "features" that describe the design for a specific task. These features contain data types such as 3D surfaces, lines, and points, as well as relationships to other features. For the arcs of this graph, we formalize a generic reasoning mechanism, called a "perspector," which analyzes any number of "source perspectives" to produce one "dependent perspective." Engineers from different disciplines use perspectors to transform source perspectives into dependent perspectives that are useful for their tasks. Dependent perspectives serve as source perspectives for other dependent perspectives, leading to a self-organizing graph of dependencies between perspectives. We describe this approach with two multidisciplinary engineering problems from the Walt Disney Concert Hall (WDCH). Perspectors and perspectives enable engineers to use design representations that share a common theoretical foundation. They allow engineers to automatically generate task-specific representations from representations produced by other engineers.

## 1 Introduction

When designing and executing a multidisciplinary project, architects and engineers produce and integrate graphic and symbolic project representations. While producing these representations, they use information contained in the representations of other disciplines. Design is an iterative process: engineers routinely modify "source" representations throughout the design process, but without being able to fully integrate their work with other disciplines on a daily basis. Engineers responsible for "dependent" representations therefore need to maintain consistency with changes made to their "source" representations. Current practice for producing and integrating task specific project representations is error-prone and costly.

**Example 1.1- Engineers produce dependent representations from source representations:** *A metal decking contractor receives 3D representations of the project's slabs and structural members (source representations) from the project's architects and steel fabricators. He generates a metal deck attachment representation describing where to install specific types of attachments that connect metal decking for concrete floor slabs to the structural beams. When he is finished, cost estimators, fabricators, and field installers use this metal decking attachment representation to produce other representations that help them further plan and execute thei project.*

*As the architect coordinates with design consultants and subcontractors, the slab and beam representations are iteratively modified, generating new metal decking attachment conditions. The metal decking contractor must notice and annotate these new conditions in the metal deck attachment representation. Representations that used information in the metal deck attachment representation must also be updated.*

This example illustrates a useful and recurring pattern in multidisciplinary design and construction projects: Dependent representations depend on source representations. We formalize a project modeling approach that exploits this pattern in a way that enables engineers to automatically generate and integrate task-specific representations. In example 1, the metal-decking subcontractor's job is to construct the "dependent" metal-deck attachment representation with information in the "source" concrete and steel representations. He applies reasoning involving knowledge about his discipline to these source representations to produce a dependent representation. We formalize the dependence between representations as:

$$\boldsymbol{R}_d = \boldsymbol{F}_r\,(\boldsymbol{R}_S)$$

$\boldsymbol{R}_d$: dependent representation
$\boldsymbol{F}_r$: reasoning function
$\boldsymbol{R}_S$: one or more source representations

The dependent representation serves as a source representation for other dependent representations. A graph of dependencies is formed. The entire collection of project representations constitutes a project plan. See Figure 1.

In traditional document-based practice, the making and updating of integrated project representations has been a time-consuming, inconsistent, and error-prone practice. When source representations are modified, designers responsible for dependent representations must become aware of these modifications, and manually represent any implications of the modification in the dependent representations.

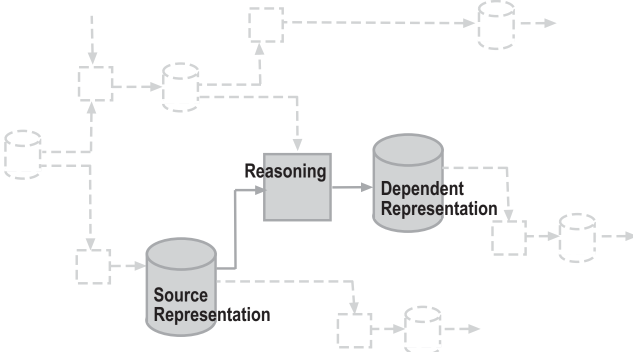



Figure 1: We conceptualize a project plan as a directed acyclic graph of representations. Engineers use discipline specific reasoning to transform source representations produced by other disciplines into task-specific dependent representations.

3D project model databases promise to automate the process of generating integrated dependent representations of changing source representations. At the beginning of a project, a project team adopts a schema that contains the specified representation conventions that they believe will be useful to the project. Standards, such as the Industry Foundation Classes (IFC 2002), are emerging to make these schemas consistent across projects and throughout the industry. Professionals add design information to the project model according to this schema, and use query languages, such as Structured Query Language (SQL 2002), to transform the information in the project model (the source representation) into useful views (dependent representations). However, engineers currently are not able to formalize much of their discipline reasoning into queries that quickly and systematically produce many kinds of explicit, accurate, and up-to-date task-specific project representations. Single, project-wide schemas do not provide individual engineers with the representations of concepts they need for specific tasks. Multiple, discipline-specific schemas lack computational support to automatically integrate these representations.

This research formalizes a computational framework for the generation and integration of a network of task-specific project representations. In this framework, we call a representation a ”perspective.” A perspective contains project information in the form of features, which can contain 3D geometry or other information. Designers use features in one or more source perspectives to produce a dependent perspective with its own features. The dependent perspective can then serve as a source perspective for subsequent dependent perspectives. In this way we conceptualize a project model as a self-organizing, directed acyclic graph of perspectives.

At a minimum, formalizing a project model in this way allows source perspectives to notify dependent perspectives when they have been modified. However, this framework also formalizes modular reasoning mechanisms, which we call “perspectors,” to automate the transformation of source perspectives into dependent perspectives. A perspector inspects the 3D features of source perspectives, and produces a dependent perspective that contains new features, describing the project plan for particular criteria. Features in dependent perspectives contain relationships to features in source perspectives, providing the expressive power to represent many types of components, attributes, and relationships such as specification, association, and aggregation. Engineers can generate new dependent perspectives at any time by specifying their dependence on source perspectives using either predefined or user-defined perspectors. Perspectors can either use automated reasoning or require user input, to produce the dependent perspective. Because of the nature of our formalism, in which a perspector depend on perspectives, to produce other perspectives, any one perspective can be defined as a directed graph of any number of lower level perspectors (See Figure 4.). This allows for encapsulation of this complexity into higher-level perspectors.

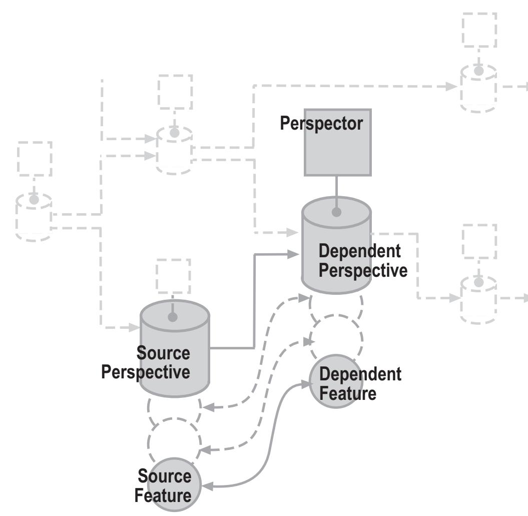



Figure 2. A perspective contains features that describe geometric aspects of a design with 3-D surfaces, lines, points, and relationships to other features. A perspector analyzes any number of source perspectives to produce one dependent perspective. Engineers from different disciplines use perspectors to define how their perspectives depend on other perspectives. The result is a self-organizing, directed acyclic graph of self-integrating project representations, which we call an integrated project plan.

In the next section we present a test case from the WDCH that illustrates the difficulties engineers currently

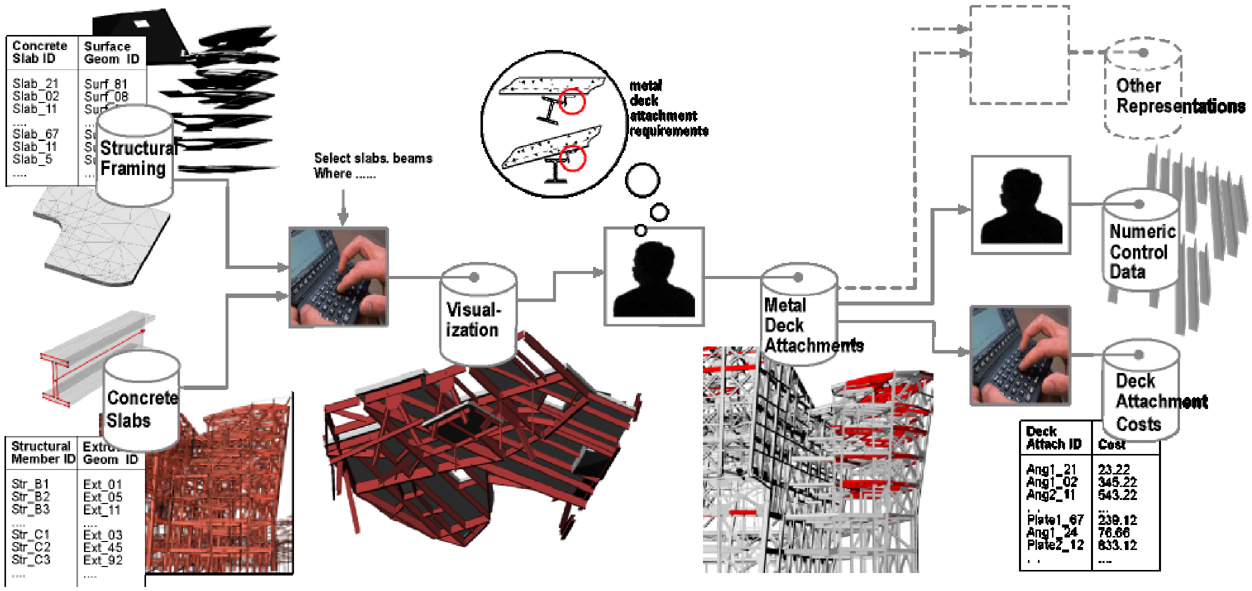



Figure 3. In current practice, designers overlay 3-D visualizations and inspect source representations (here 3-D models of the steel and floor slabs) to manually create a new dependent representation describing which beams require deck attachment to connect to the slabs. The metal decking attachment representation is then used by other disciplines.

face while constructing and maintaining integrated project representations from state of the art project models. We then discuss research in the areas of project model representation and reasoning, identifying how this research helps but, alone, does not appease these difficulties. Next we further elaborate on perspectives and perspectors, illustrating how they can be used to address the test case. Finally we look at the limitations, implications, and future possibilities for perspectives and perspectors.

## 2 Test Case: The Walt Disney Concert Hall

**Example 2.1- Engineers can automatically generate some dependent representations:** *The architect produces a 3D model of the concrete slabs. He represents each slab with a 3D surface. The structural fabricator produces a 3D model of the structural steel as a polygon that describes the cross-section and a point-vector that describes the location, orientation, and length of extrusion. A cost estimator with reasonable knowledge of SQL can construct a query to calculate the approximate cost of metal decking on the job, even if the metal decking is not explicitly represented in the project model. In other words, he can transform the source representation (a description of the slabs beams) into a dependent representation that describes the material cost of metal decking. He knows that, while metal decking is not represented in the database, wherever there is a slab, a metal deck is required, He therefore can write a query to calculate the area of slab, and calculate metal decking cost as the area of the slab multiplied by the average cost of metal decking per square foot.*

The cost estimator could manually calculate and annotate the cost for each metal deck. However, using SQL, he automatically constructs a useful dependent representation from source representations, assuring accurate, consistent, task-specific representations at a minimum time and cost. As the next example suggests, however, current practice fails to automate the generation and integration of many types of dependent representations that engineers need.

**Example 2.2- Engineers cannot automatically generate some dependent representations:** *The metal decking subcontractor needs to produce a metal decking attachment representation describing how to attach the metal deck to each of the beams (see figure 3). Constructing this representation involves comparing the spatial relationships of the slabs and beams to determine which beams support individual slabs and, in these cases, what types of connections are required. For example, if the slab is not firmly resting on the top face of the beam, custom support angles must be added.*

*There is no representation of this condition in the design data, and it is not easy to construct a query such as, "Select the top edge of the beams that are below, touching, and not parallel to the bottom face of the slabs." Therefore, the metal decking subcontractor must overlay 3D visualizations of both slab and concrete representations, annotate every with a line in each location where a custom deck support angle is needed. Such work is painstaking and error-prone. On the WDCH project this task took approximately 120 hours to complete.*

*After the initial identification of these conditions, design coordination continues. Whenever any information in the structural steel or concrete representations change (for example, if a slab or beam is added, modified, or deleted) the metal decking contractor needs to become aware of these changes, and manually update the (dependent) metal decking attachment representation. The project goes through several design iterations. In various phases of the project, the latest version of the metal deck attachment representation is useful to the estimator for determining the cost of the attachments, to the architect to coordinate slab openings, and to the fabricator to plan, fabricate, deliver, and install these angles. Each of these dependent representations must be updated whenever the metal decking attachment representation is changed. Missed conditions in this iterative transformation of information between representations result in inaccurate cost and time estimates, design conflicts, change orders, and delays in the completion of the project.*

Example 2.2 shows that existing methods fail to provide engineers with the ability to quickly and accurately generate many types of task-specific dependent representations from source representations. We therefore introduce perspectors and perspectives that together enable practitioners to automatically transform source representations into dependent representations. When source perspectives are modified, dependent perspectives can be regenerated. Figure 4 uses perspectors on the metal decking test case. In Figure 4, at position *A,* perspectors analyze the structural framing and concrete slab perspectives to generate dependent perspectives describing the top and bottom face of the beams and slabs. At position *B*, these dependent perspectives become source perspectives for perspectors that produce dependent perspectives describing relationships between these faces. At position *C,* a Perspector generates a perspective describing which beams satisfy all the relationship conditions (touching, not parallel, above). All of these perspectors are encapsulated, at position *D*, into one perspector that analyzes one concrete slab perspective and one structural framing perspective and produces one deck angle attachment perspective.

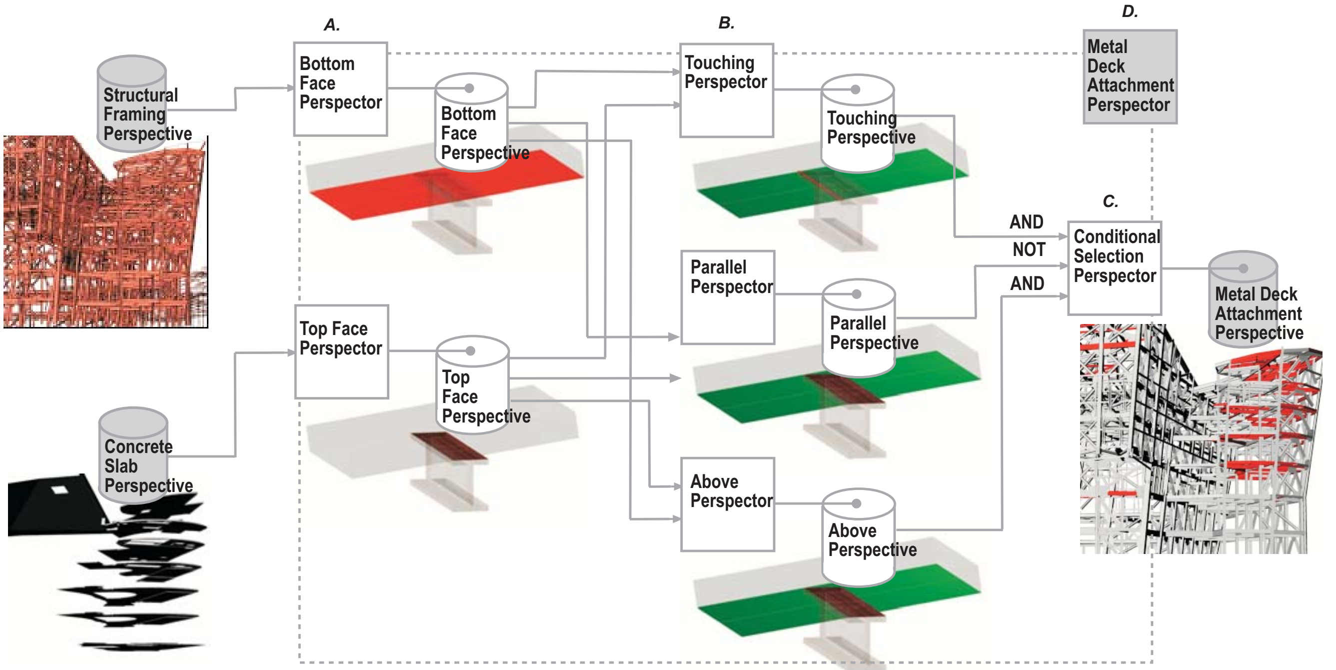



Figure 4. The deck angle attachment perspector is a combination of perspectors that identify features of slabs and beams (bottom face, top face, and top edge) and relationships between these features (touching, parallel, and above). The deck angle attachment perspector automatically generates a perspective from the structural framing and concrete slab perspectives.

## 3 Related Research

In this section we discuss related research involving the representation of project models and reasoning about these project models.

### 3.1 Relation of perspectives to prior work in project model representations

Engineers need to construct task-specific representations. Figure 5 illustrates that standardized representation approaches contain many, but not all, of the concepts required to represent the existence, length or location of the metal deck attachment. Using the IFC we represent that a surface of the beam and a surface on the slab are connected. There are no formalisms to identify which surface is the top face or to represent that the connection type is one that requires a deck angle attachment. In order to appropriately represent a deck angle attachment engineers need to extend these schemas to include concepts such as bottom face, top face, touching, parallel, above, and deck attachment.

The domain of project model representations involves defining the relevant objects, attributes, and relationships in a project model to enable information sharing among disciplines. Some of these approaches use a central shared model (IFC 2002, STEP 2002), while others use multiple, domain-specific models with integrity relationships between them (Turk 2001, Rosenman & Gero 1996, Mackellar & Peckham 1998). Some (Björk 1987, IAI 2002) explore a semantically explicit approach providing specific objects (such as a beam), attributes (stating that the beam is a W12 X 42), and relationships (stating that the beam supports the slab). Others approach the problem syntactically, providing abstract structures such as objects, attributes, and relationships that can be extended to create a particular schema (Phan 1993, Stouffs 1997, Clayton et al 1999, Van Leeuwen 1999). Some of the above approaches can be used a priori, or at design time (IAI 2002, Gielingh 1988). Others are intended to be used a posteriori, or during design inspection (Clayton 1999, Hakim & Garrett 1997).

Perspectives are a syntactic approach. They allow engineers to define new conceptual objects, attributes, and relationships in the project model at any time. We extend these syntactic approaches by incorporating reasoning in the form of perspectors, to automate the generation and integration of instances of these concepts in an existing project model.

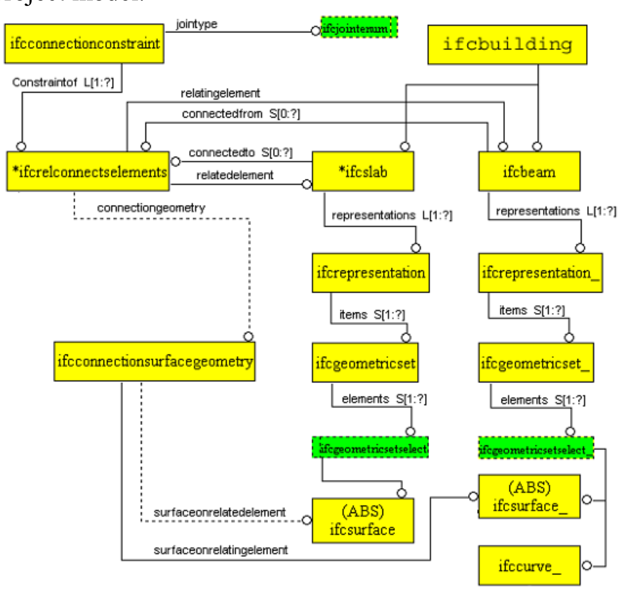



Figure 5: Here we use the IFC to attempt to represent the relationships between a slab and beam in a way that facilitates the design of deck angle supports. This representation states that a slab is connected to a beam. However, there is no representation of specific aspects of components (i.e. top face, nor of the spatial relationships between these aspects (i.e. touching, above, parallel).

### 3.2 Relation of perspectors to prior work in reasoning about project model

Most research involving reasoning about project models has been done in the context of a single discipline, or between disciplines. This work involves generating instances of objects, attributes, and relationships for specific tasks. This work is useful because it specifies many types of reasoning and representations required for task-specific criteria, which can be implemented as perspectors and perspectives. Among some projects at CIFE: Darwiche et al (1988) perform model-based reasoning to produce a construction schedule; Akinci (2000) analyzes a 4D model to infer time-space conflicts for workspaces; Akbas et al (2001) analyze project geometry with productivity constraints to determine daily work zones; Fischer (1993) analyzes project models for constructability concerns; Han et al (2000) analyze an IFC-based project model for handicapped accessibility; Korman et al (2001) perform MEP coordination, and Staub-French et al (2002) formalizes the automation of cost analysis. Outside of CIFE many others have created similar model-based reasoning systems. Dym et al (1988) performs automated architectural code checking. Others focus on performing a series of tasks around an integrated project model (Aouad et al 1997). These systems require a fully developed explicit building project model schema and instance, which represents all of the required information for each discipline. Reasoning transforms instances of the source schema into instances in the target schema. These systems are not designed to allow engineers to construct new instances of new concepts.

Query languages like SQL enable the automatic transformation of source representations into dependent representations that contain instances of new concepts. However, the test cases suggests that existing query languages are not suited to multidisciplinary design, and therefore are not used broadly in either practice or research. This is in part there because SQL does not define many of the transformations designers would find useful, and in part because there is not a framework to facilitate the assembly and management of transformations to generate integrated representations for multidisciplinary teams. Other syntactic approaches to reasoning about project models (IAI 2002, Eastman 1995) define constraints between objects that include tests for validity of these constraints. For example, Figure 5 illustrates an "ifcconnectionconstraint" that monitors the slab and beam, assuring that their attributes - such as their geometric descriptions - match predefined criteria. If the constraint is violated, the relationship is terminated. However, these constraints are applied in an a priori fashion as part of the concept definition, not allowing for a posteriori automated analysis of the implicit conditions in the existing project model.

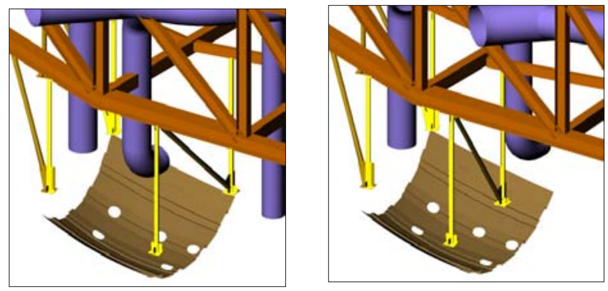
In the mechanical engineering domain, many have worked "feature recognition". (Rosen et al 1994, Mantyla et al 1996). For example, Rosen and Dixon formalize feature recognition as a process of filtration, annotation, and aggregation, allowing for the detection of ‘primitive features’ from component geometry. Most of this work is primarily focused on analyzing component features aspects of those components. Summers et al (2002) formalize a “design exemplar,” in which they use generic representation of entities and their relationships, evaluating a database of design cases using a general algorithm based on constraint satisfaction to search for cases that match a defined “exemplar.” Wilson et al (1995) reason about how an assembly of mechanical components can be disassembled, using what they call a “non-directional blocking graph.” They geometrically analyze the configuration of components, and then organize the components in a graph that describes the order in which the components can be removed.

Prior research provides useful points of departure in terms of the specific representations and reasoning that are useful for specific disciplines. We formalize perspectors and perspectives that allow engineers to generate a self-organizing, integrated project model, and thus allow them to define their representations as a collection of transformations of other representations.

## 4 The Perspector Framework

We formalize generic reasoning and representation that is arranged in a graph to automate the construction of dependent representations from source representations. We explain these concepts in the context of another WDCH test case.

**Example 4.1: Engineers can use perspectors to automatically generate dependent perspectives:** *Architects, engineers, contractors, and subcontractors all collaboratively design the ceiling system of the WDCH. Ducts, catwalks, fire sprinklers, theater lighting, and several other systems vie for a tight space above 200 3m x 4m ceiling panels that weigh in excess of 1 ton 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. 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, keeping these cantilever conditions to a minimum is desirable.*



**a** Panel with edge supports **b** Panel with two cantilevers

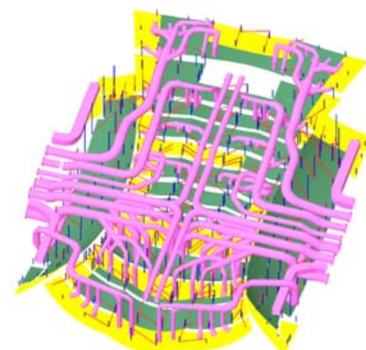


Figure 6 The WDCH ceiling, consisting of over 200 panels, from above. Panels with cantilever supports are highlighted. The ductwork is overlaid. Insets **a** and **b** show individual panels, with connection to steel hangers that are connected to roof trusses above. Only the bottom surface of the panel, as the panel framing has not been designed. The hangers shown in **a** are directly over the corners of the panel, therefore there is no cantilever. In **b** two hangers were moved to make room for a duct, creating two cantilever conditions.

*An engineer responsible for addressing these conditions could, in a user-interface as shown in figure 7, assembles perspectors like those shown in figure 8. In this perspector graph, he wants to determine which supports are supporting which panels, and of these, which are in cantilever position. To do this, he converts each panel feature (which contains a triangular mesh representing the surface of the ceiling panel) into a feature that represents the edges of the panel, with a poly line describing this edge. He then converts each hanger feature into a feature with one point representing the center point of the hanger. Next, he determines which hanger center points are inside which panel poly line. The result is a representation of which hangers are supporting which panels. The final step is to measure the shortest two distances of each of these* *points to each of the edges of the polygon. If this distance is greater than a user defined value, this condition classifies as a cantilever. There are probably several ways to describe a cantilever condition, but if standardization is required, one algorithm can be specified.*

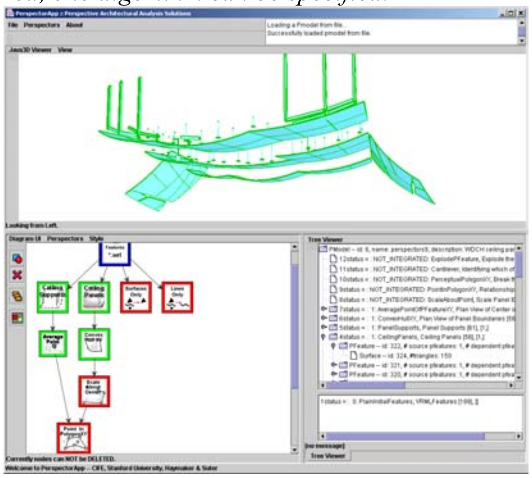


Figure 7: A prototype: engineers arrange a graph of perspectors in the lower left window. They can navigate through a 3D view of the features of any selected perspectives in the top window. They can traverse a tree view of all perspectives, features and the data of features in the lower right window.

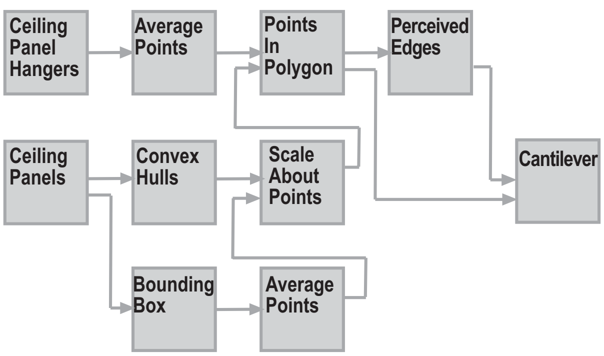



Figure 8: A perspector graph that constructs a perspective describing cantilever conditions.

### 4.1 Perspectives: Generic concepts

A perspective has references to its source perspectives and contains any number of features. A perspective also has a reference to its perspector and to it's status, which states whether this perspective has been integrated with its source perspectives. The semantics of the perspective determine whether to interpret its features as instances of components, attributes, or relationships.

The "PointInPolygon" perspective is represented as:

**Name**: pv_HangerSupportsCeilingPanel_01
**Description**: Each feature contains a polygon, which represents the edges of a "ceiling panel," and any number of points, each of which represents a "hanger".
**Perspector**: cife.perspector.geom..PointInPolygonXY
**Source Perspectives**:pv_AvgPnt01, pv_SclAbtPnt01;
**Status**: integrated
**Features**: pf_PntInPgonXY_01,
pf_PntInPgonXY_02

### 4.2 Features: Generic instances of a concept

A feature has source features, dependent features, and is used to describe a concept in terms of any number of surfaces, lines, or points or other data types. Through recursive source and dependent features, engineers can construct complex concepts such as objects (a ceiling panel feature), relationships (The hangers support the ceiling panel), and attributes (the edges of the panel). For example, perspector cife.perspector.geom.PointInPolygonXY represents one of it's features in this context as:

**Name**: f_Supp_01
**Perspective**:pv_Support_01
**Sourcefeatures**:f_AvPt1,f_AvPt7,f_AvPt2,f_SclAbtPt4;
**Surfaces**: null
**Lines**: line01
**Points**: point02, point05, point23, point21, point02

### 4.3 Perspectors: Generic reasoning

A perspector encodes the reasoning to analyze features in source perspectives to produce features in the dependent perspective. This idea is applied recursively, allowing for complex, nested, reasoning. Perspectives pass parameters to perspectors allowing more generality for each perspector. For example, the scaleAboutPoint perspector accepts a parameter to specify how much to scale the geometry in the feature.

The PointInPolygon perspector takes two perspectives as source perspectives. The algorithm takes the first line in each feature of the first perspective, and compares it to every point in every feature of the second perspective. The perspector returns a dependent feature in the dependent perspective for every line it analyzes. This feature contains this line, and every point that is inside this line. Such a feature is represented in Section 4.2.

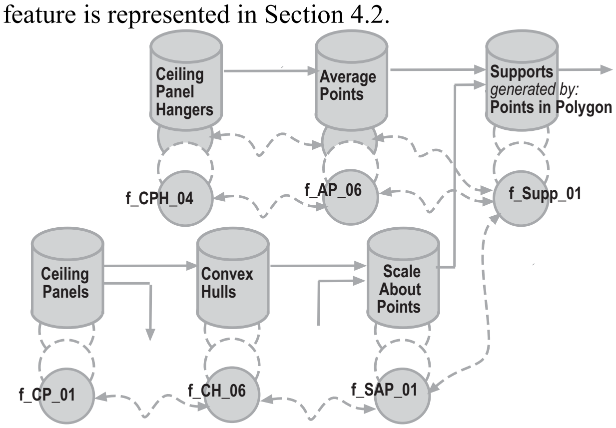


Figure 9 A PointInPolygon feature (f_Supp_01) relates what points are in each polygon. Through the source feature relationships, this feature also represents which ceiling panel hangers support which ceiling panels. Therefore, the PointsInPolygon perspective and its features can be renamed, “supports,” in the context of the graph. Two hangers are shown supporting one panel. This feature is then analyzed to determine which supports are cantilevered.

# 5 Limitations, Future Work, Implications

We present a syntactic framework in which engineers define dependent representations as a graph of dependencies on source representations. The approach provides integrated, task-specific representations of an evolving design. The examples given involve representing the spatial relationships between building components in useful ways on a multidisciplinary building design and construction project.

Limitations of the approach may stem from the formalization of the dependence between representations as a directed acyclic graph. The implication is that dependent representations, and the reasoning used to create dependent representations from source representation, have no effect on these source representations. While the usefulness of this abstraction has yet to be fully validated, the test cases provide evidence that formalizing the dependence as an acyclic graph provides the ability to generate dependent representations more quickly, accurately, and systematically than without them. Another potential limitation is that many types of discipline reasoning may prove difficult to formalize using perspectors. The framework allows for both manual and automated generation of perspectives. A perspector can simply ask the user for input. A perspector graph with just user input perspectors is still valuable. It can be used to notify dependent perspectives that the source perspectives have been modified.

Future work involves developing specific feature representations and specific perspectors for different types of multidisciplinary design problems. Doing so will further the understanding of the types of representations that can be automated using this approach and the types of reasoning needed to construct these representations. It is possible that a finite set of different types of perspectors could emerge, allowing the systematic specification of representations. Understanding the project needs for management of the graph, developing a more intuitive user interface for constructing these graphs, and working with non-geometric data in features are areas of future work.

Implications of the approach lie in the systematic specification of representations through their dependence on other representations. Generating dependent representations from source representations may be of interest to multicriteria automated design systems, and to any application that needs multiple, task-specific representations of a changing 3D scene. Finally, the process of generating dependent representations from source representations can be used for design generation.

# 6 References


Ragip R. Akbas, Martin A. Fischer, John C. Kunz (2001). “Formalizing Domain Knowledge for Construction Zone Generation.” Proceedings of the CIB-W78 International Conference IT in Construction in Africa 2001, Pretoria, South Africa, pp. 30-1 to 30-16.

Akinci, B., Fischer, M., Levitt, R., Carlson, B. (2002). "Formalization and Automation of Time-Space Conflict Analysis." Journal of Computing in Civil Engineering, ASCE.

Aouad. G, Marir. F, Child. T, Brandon. P & Kawooya. A. Construction Integrated Databases- Linking design, planning and estimating. Proceedings of the international conference on the rehabilitation and development of civil engineering infrastructures. American University of Beirut, June 1997, pp 51-60.

Bjork B-C (1987). RATAS: A proposed Finnish building product model, Studies in Environmental Research No. T6, Helsinki University of Technology, Otaneimi, Finland.

Clayton, M. J., Teicholz, P., Fischer, M.A., Kunz J.C., “Virtual components consisting of form, function, and behavior”, 1999, Automation in Construction, 8, 351-367,

Darwiche, A., Levitt, R.E., and Hayes-Roth, B. “Oarplan: Generating project plans in a blackboard system by reasoning about objects, actions, and resources.” Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2(3):169-181, 1988.

Dym, C.L; Henchey, R.P.; and Gonick, S. (1988). "A Knowledge-based System for Automated Architectural Code Checking." Computer Aided Design, 20(3), 137-145.

Eastman, C., Jeng, T-S., Assal, H., Cho, M. and Chase, S. EDM-2 Reference Manual, Center for Design and Computation, UCLA, Los Angeles, USA, 50pp, 1995

Fischer, Martin A., Automating Constructability Reasoning with a Geometrical and Topological Project Model, 1993, Computing Systems in Engineering, 4(2-3), 179-192.

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

IAI (2002), “Industry Foundation Classes, Version 2.X”, International Alliance for Interoperability

Han, C.S., Law, K., Kunz J.: "Computer Models and Methods for a Disabled Access Analysis Design Environment", Technical Report Nr 123, CIFE, Stanford University (2000)

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

Korman, Thomas M., Tatum C.B. (2001) “Development of a Knowledge-Based System to Improve Mechanical, Electrical, and Plumbing Coordination" Technical Report Nr 129, CIFE, Stanford University

Leeuwen, J.P. van. 1999. Modelling Architectural Design Information by Features. PhD. Thesis. Eindhoven, NL: Eindhoven University of Technology.

MacKellar, B. & Peckam, J., ‘Multiple Perspectives of Design Objects’, Artificial Intelligence in Design ’98, ed. John Gero and Fay Sudweeks, Klewer Academic Publishers (1998) pp. 87 – 106

Mantyla, M., Nau, D., Shah, J., (1996) 'Research Challenges in Feature Based Manufacturing', Communications of ACM, Feb.

Phan, D.H. Douglas, Howard, H. Craig (1983) "The Primitive-Composite (P-C) Approach: A Methodology for Developing Sharable Object Oriented Data Representations for Facility Engineering Integration" Technical Report Nr 85, CIFE, Stanford University

Rosenman, M. A. and Gero, J. S. ‘Modeling multiple views of design objects in a collaborative CAD environment’, CAD, Special Issue on AI in Design 28(3) (1996) pp 207-21

Staub-French, Sheryl, and Fischer, Martin A. “Formalisms and mechanisms needed to maintain cost estimates based on an IFC product model”, 2000, Stanford Univ., Eighth International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII), Renate Fruchter, Feniosky Pena-Mora and W.M. Kim Roddis (Eds.), 716

STEP (1999), ISO 10303, Standard for the Exchange of Product Model Data

Stouffs R. and Krishnamurti R., Sorts: a concept for representational flexibility, CAAD Futures 1997 (ed. R. Junge), pp.553-564, Kluwer Academic, Dordrecht, The Netherlands

Summers, J., Lacroix, Z., Shah, J., 2002, "Case-Based Design Facilitated by the Design Exemplar", Seventh International Conference on Artificial Intelligence in Design, '02, ed. J. Gero, Kluwer Academic Press, Netherlands, pp. 453-76, ISBN 1-4020-0716-7.

Turk, Z. (2001). "Phenomenological foundations of conceptual product modeling in architecture, engineering and construction." Artificial Intelligence in Engineering; 15(2), 83-92.

**Acknowledgements**:

We appreciate the support of the Center for Integrated Facility Engineering at Stanford University, M.A. Mortenson Company, Frank Ghery and Associates and Walt Disney Imagineering.