Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2000), *14*, 137–147. Printed in the USA.
Copyright © 2000 Cambridge University Press 0890-0604/00 $12.50

Integrating product models with engineering analysis applications: Two case studies

J. ANDREW ARNOLD AND JOHN C. KUNZ

Center for Integrated Facility Engineering, Stanford University, Stanford, CA 94305, USA

(RECEIVED May 12, 1999; REVISED August 20, 1999; ACCEPTED August 31, 1999)

Abstract

Current methods to develop standard Architecture, Engineering, Construction (AEC) product models focus on the definition of product model semantics without concurrent and formal consideration of the engineering analyses that such models must support, or formal consideration of the requirements for sharing information between applications. We present two case studies that demonstrate a service to extract data from product models and provide inputs to component analysis applications. The service was validated in a proof-of-concept application called the Internet Broker for Engineering Services (IBES) that extracts information for component analysis from product models that are external to the application and accessed across the Internet. IBES was tested for two research cases. The product model for the first case, control valve selection is based on STEP Application Protocol 227. The product model for the second case, control valve diagnosis, specifies additional semantics that support the operations and maintenance (O&M) phase of the facility life cycle. The cases offer evidence that large standard data models can support routine analyses for control valves. However, the amount of shared information between the case applications is small and is largely dependent upon the concurrence of component behaviors that are necessary to model analysis. The IBES reference model and reasoning to support information extraction was consistent for both cases. This consistency suggests that it is possible to define a general set of computational methods that integrate project information models with external component analysis applications across the product life cycle. We argue that enabling a web-based link between product models and applications requires a set of capabilities, including bi-directional communication between separated data and analysis nodes, query generation, data translation, and validation of data extracted from semistandard models. We discuss the tentative implication that minimal shared information calls into question the assumption that large core product models will work effectively in practice.

Keywords: Architecture; Engineering; Construction; Product Models; Engineering Analysis; IBES; Integration Service

1. INTRODUCTION

The Architecture, Engineering, Construction (AEC) industry is changing as the complexity, costs, and risks associated with project delivery drive the search for business practices and technology that increase the speed of production and reduce the cost of new facilities.

For example, consider component selection for a process plant. Contractors and component vendors need to identify component configuration and installation methods as early as possible to reduce production risks. Often component vendors perform final component specification, contingent upon the quality of information received from the engineer. However, the design engineer frequently postpones final component specification to accommodate late design and equipment changes. Traditional methods "push" engineering information from the engineer to the vendor. The conflicting pressures result in suboptimal coordination, errors, design data duplication, re-work, and ultimately, poor performance.

Industry initiatives such as STEP define product description standards. These standard models assume that a shared *core model* will enable AEC application integration. A core model represents products from multiple engineering disci-

Reprint requests to: J. Andrew Arnold, Center for Integrated Facility Engineering, Stanford University, Stanford, CA 94305, USA.

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

plines and thereby enables multiple engineering analyses using the same product model. For example, a core model enables detailed design and subsequent operations-phase diagnosis of a component such as a pump. Thus, a shared core model would make it possible to "pull" vendor expertise and information on demand for engineering analyses across the facility life cycle. The Internet enables such sharing among information providers and consumers. Our research presents preliminary empirical evidence for characterizing the information sharing requirements between like and unlike AEC engineering analysis applications. Our intent is to define useful strategies that enable AEC application integration.

Figure 1 illustrates our view of the process of brokering engineering content between standard data models and specific requirements for analysis.

Efforts to define standard product models in the AEC industry expend considerable resources on the development of large core product models; however, there is little evidence to indicate what information is shared and what is distinct between applications. This paper examines the information requirements for two component analysis applications for the process industries, preliminary valve selection and diagnosis. We developed a systematic method to extract project data for use by component analysis applications, and found that in fact, few data items were shared between the two applications.

We developed the Internet Broker for Engineering Services (IBES) prototype application to automate the process for this research. It systematically extracts project data from remote component and plant models to enable their use by component analysis applications. IBES extracts design data (Fig. 1A.) for the valve-sizing case. It extracts design data and valve assembly information (Fig. 1B) from an operations and maintenance (O&M) library of components for a valve diagnosis application. Diagnosis is required during the O&M phase of the product life cycle to determine maintenance and repair requirements. We found that few data items were shared between the two cases.

2. BACKGROUND

The engineering problems of developing design details and diagnosis require multiple kinds of information:

- Form: Component features (i.e., valve-closing mechanism), topology, physical layout, material, and for process plants, the stream being carried by components (e.g., steam, water);
- Function: the engineering function or design intent of the component in the facility (e.g., pumping) and the function of the engineering process (e.g., valve sizing and selection, diagnosis, or data extraction).
- Behavior: the measured or computable properties of the components (e.g., predicted valve inlet pressure, coefficient of flow). For most engineering processes, it is useful to distinguish between predicted and measured values of behaviors (e.g., nominal vs. actual valve actuator travel).

Generally, AEC efforts to define product models prescribe component semantics in specialization hierarchies (Bjork, 1989; Amor et al., 1993). Most approaches attempt to define a shareable core model for extension by different AEC disciplines. Other approaches investigate evolutionary product model development (Eastman & Sheng Jeng, 1998) or interactive CAD model annotation as an outgrowth of cognitive design processes (Clayton et al., 1994). Certain prescriptive models are sufficiently mature to support engineering applications. For example, STEP for the process industry has produced two Application Protocols (AP) 221, functional design description (ISO, 1997) and AP 227 physical and logical plant layout (ISO, 1994). A committee draft version of AP 227 provides the data model for the first case described in this paper.

Standardization efforts, for example STEP (URL: http://www.mel.nist.gov/sc5/soap/) and IFC (URL: http://www.interoperability.com/), describe form, and they provide some description of function. Normally, they provide little description of behavior, and they make no standard provi-

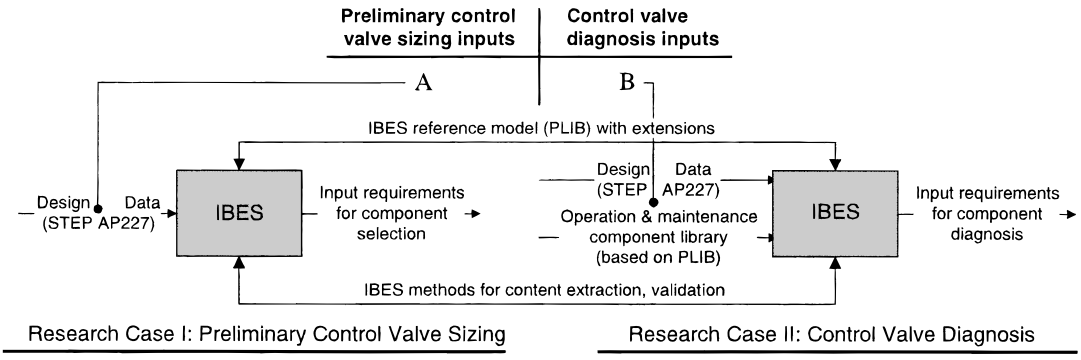


Fig. 1. Source information shared between analysis applications.

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

sion for defining computational methods to compute predicted behaviors. The two analysis examples we describe each require appropriate specification of form, function, and behavior.

The following section describes the control valve-sizing and diagnosis test cases.

3. PRELIMINARY VALVE SIZING

The control valve-sizing case uses a textbook problem for selecting the correct valve size under conditions of turbulent liquid flow.

> An eight-inch schedule 30 pipe has a flow of a maximum 1600 gallons per minute (gpm) of river water. The pressure immediately upstream of the control valve is 27.9 pounds per square inch gage (psig), or 42.6 pounds per square inch actual (psia). The pressure in the pipe immediately downstream of the valve is 20 psig, or 34.7 psia (Driskell, 1983).

The engineering task is to determine the required control capacity behavior for the valve by computing the coefficient of flow (ISA, 1991), $q = CvFp\sqrt{\Delta p/G}$, from the given conditions and subsequently to select a control valve that meets the requirements. If a design engineer wants to use a 60-degree butterfly valve, a 6″ nominal diameter valve is satisfactory. The solution is obtained by entering the design condition variables into a product selection program provided by a publisher, or by using the equation to compute flow manually and inspecting the proper selection table in a publisher's product catalog. An engineer typically calculates the *Cv* values for the minimum and maximum process control conditions.

4. CONTROL VALVE DIAGNOSIS

The second case, control valve diagnosis, was developed from a technical paper provided by a control valve vendor (Fitzgerald, 1990). The engineering task is to analyze and evaluate valve behavior during plant operation and recommend a maintenance and repair plan. The diagnostic method compares current valve assembly behavior with historical performance behaviors. This procedure, called signature analysis, ". . . measures valve stem travel in response to the instrument signal while simultaneously measuring instrument air to the positioner, supply pressure and actuator pressure" (Fitzgerald, 1990). From these measures, any two of the following parameters can be analyzed and compared to an optimal case (Fitzgerald, 1990):

- Actuator spring rate and bench set;
- Valve stroke and friction;
- Stroking speed;
- Input signal to pressure transducer calibration linearity and hysteresis;
- Packing friction;
- Stem-nut adjustment;
- Air supply pressure;
- Positioner calibration, linearity, and hysteresis;
- Seat load.

The research case application evaluates a subset of these factors, specifically, valve stem travel and friction, bench set load, and the valve plug seat load for balance and unbalance forces. It relates variant behavior to the degradation of valve assembly parts, and recommends maintenance procedures for valve stem packing, adjustment of the linkages, and calibration of the input signal and seat loads. The analysis application was tested for two examples of valve performance that were described in the vendor technical paper (Fitzgerald, 1990). One example described nominal valve performance, while the other described a valve with operating problems.

For example, Table 1 provides nominal and actual values for valve component behaviors, and diagnosis heuristics for a 2″ globe valve with an air-to-close actuator, flowing up, that is performing suboptimally (Fitzgerald, 1990):

Given the diagnosis, the repair recommendation is to take the valve off-line for disassembly, repack the stem, readjust the linkages, and recalibrate the input signal and the loads for new bench set.

5. SOURCE MODELS

As noted above, the first case used component form and function information extracted from a product model based on STEP AP227. The second case used information from AP227 and a second model that represented additional O&M component operation behaviors. The following section describes these models.

5.1. Case 1 data model

The first prototype integrates a CAD representation of a process and instrumentation diagram (P&ID) with an underlying product model. The product model structure is based on the object model, or Application Reference Model (ARM), of the STEP standard for plant spatial configuration, (AP227) (ISO, 1994). The intended scope of AP227 is the physical and logical layout of process plant components and systems to support component interference analysis. The model also contains some functional design information. Nominally, ISO STEP AP221 (1997) represents functional design data, however AP221 was not available when this research began. The model for the case adds a *piping_specification* class for *pipe_class* and *pipe_material* properties (Fig. 2) to support product model interpretation for valve sizing.

Figure 2 shows part of the AP227 model. It highlights structural redundancy and many levels of indirection between the Start node (A) and Goal node (B) for the research case query that extracts information for valve sizing. Links

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

Table 1. *Nominal and actual conditions for a 2″ globe valve with an air-to-close actuator, flowing up, that is performing suboptimally*

Component Behavior	Nominal Value	Actual Value	Diagnosis
Actuator stem travel	0.75″	0.86″	The stem is over traveling by 14%. Because the valve is plug characterized, little additional flow is gained. This portion of the Input signal is wasted. There is also a risk of putting the stem connection in tension by over stroking to the point where the top of the plug could interfere with the bottom of the bonnet.
Valve friction	320 lb	238 lb	The valve friction is lower than the nominal. The packing might be leaking.
Bench set	3–17 psi	5.9–26.2 psi	Bench set values are higher than nominal. The spring rate, Max: 1403 lb, is too tight. It absorbs more of the air pressure than it should during the closing stroke. The maximum bench set value, 26.2 psi, results in 632 lb seat load when the valve is fully closed.
Plug seat load		Seat load: 632 lb. Unbalance force: 794 lb.	Unbalance force exceeds seat force. This condition causes the plug to bounce on the seat, damaging the valve body.

between objects represent named relationships. Thick line weights indicate the path used by the test case query to link the Start node to the Goal. The indirection requires multiple database query joins to link the Start to the Goal. Currently, developers write usage guides to resolve implementation ambiguity caused by redundancy.

The data requirements of the cases necessitate consistent methods to extract information from the source model. In the presence of redundancy, and hence ambiguity in the models, automation requires standardization of the query procedure. In addition, the large number of database joins that are necessary for a query to succeed introduce the possibility that models have incomplete information. We explicitly implemented a data validation process to check for successful data extraction. For this research, validation denotes pattern matching data definitions and values from views of data extracted by query from a product model with the input conditions for a component analysis application.

The query for valve sizing links the *plant_item* topology, representing logical plant components, to the *plant* topology, representing component function. To begin a query, the user selects a control valve symbol from the CAD P&ID that serves as the front end for the model. The graphic model object links to a symbolic representation, for example, *valve* in Figure 2.

The system presents a dialogue box list of engineering services that are registered in a component library based on the ISO Parts Library Standard (PLIB) (ISO, 1995) for the valve class. When the user selects preliminary valve sizing, the system calls a query, given the service and schema of the product model. The query searches the model of the pipe circuit upstream and downstream of the valve (Start condition) for pipe components. Then it searches for the associated functional design information (Goal condition) that supports preliminary sizing. A *valve* associates with *straight_pipe* by database joins to *plant_item_connection_occurrence.* Alternatively, the model also describes piping circuits in the functional topology, *Pipe* instances also associate with functional design information object instances via joins through *line_less_ piping_system.* Alternatively, *pipe* associates with functional information by a database join through *line_ piping_system_component_assignment.*

5.2. Case 2 data model

The O&M model for Case 2 required a detailed description of features of the control valve design (valve body, actuator, positioner, etc.), shown schematically in Figure 3, so that operation behaviors of the parts could be measured and derived for diagnosis. Using PLIB modeling resources, shown diagrammatically in Figure 4, we built an object-oriented symbolic model to describe the valve assembly in an O&M library of components that comprise a plant system.

5.3. Comparison of input conditions

Table 2 lists the properties that describe the valve assembly for the diagnosis case and compares them with the functional design properties for preliminary valve sizing. The research case analyses share only the logical identifier and valve class (style) properties that IBES extracts from several large data models. The valve class implicitly describes the type of process control that it provides. There is an approximate correspondence between inlet and outlet pressure for minimum, maximum, and upset operating conditions in Case 1 with inlet and outlet pressure for a fully open and fully closed valve in Case 2. However, they were categorized separately because their values would be different for a given sampling.

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

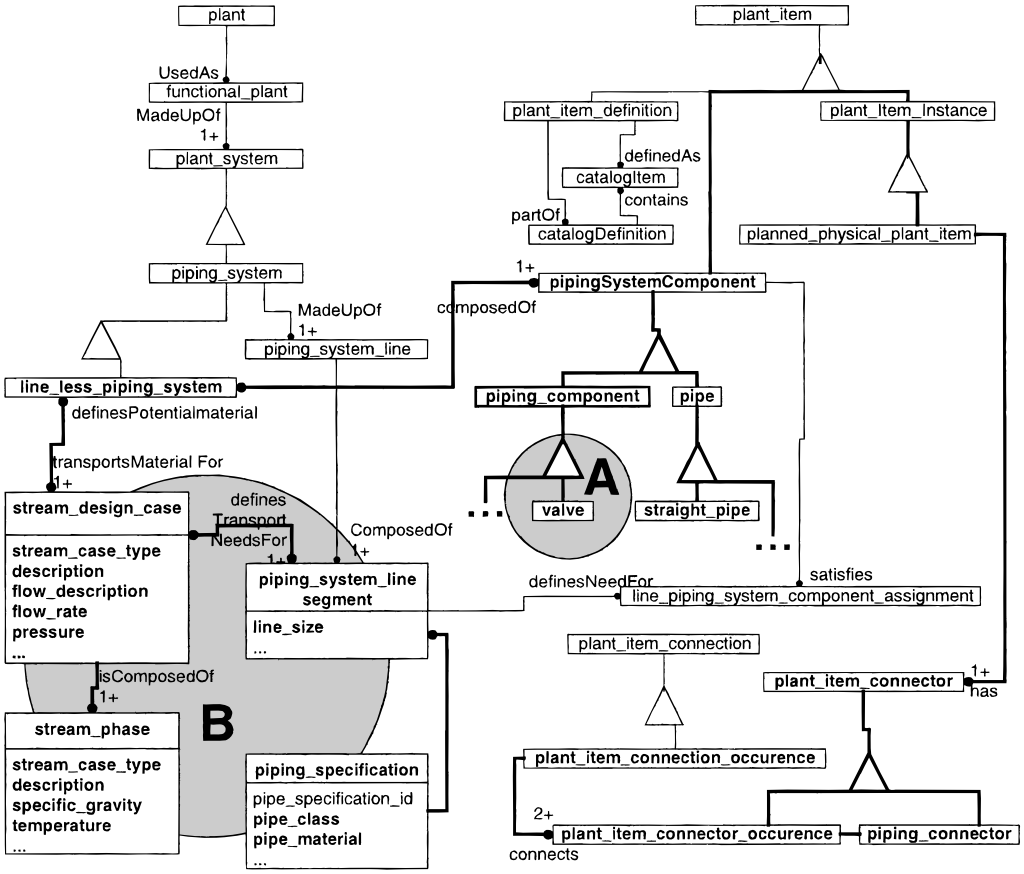


Fig. 2. AP227 model fragment (OMT notation).

Both the predicted and measured component behaviors are disjoint for the two analyses described in this paper. Similarly, the computational methods for analysis require distinct descriptions of project information, for example, Inlet pressure for valve sizing versus Inlet pressure, fully open and closed valve for diagnosis. Finally, the analysis algorithms themselves are clearly distinct.

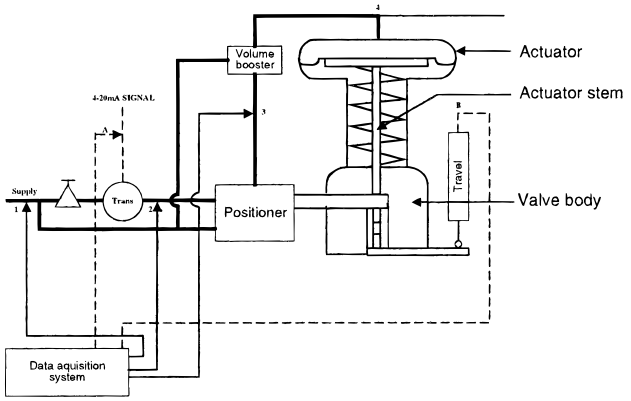


Fig. 3. Schematic valve assembly (Fitzgerald, 1990).

6. BROKERING ENGINEERING CONTENT

The IBES service dynamically extracts and validates (i.e., customizes) relevant information from product models to use as input to component analysis for both applications (see Fig. 1).

The brokering service evolved from the observation that component analysis input conditions could constrain the search for relevant data. To support reasoning for information extraction in IBES, we added templates (see Fig. 5) and control objects to the PLIB modeling resources. A template represents a class of engineering analysis function, for example, size a valve. Each attribute that describes a template class has structure, normally including name, unit, and datatype. Templates, query, and translation rules map standard product model instances (e.g., STEP) to an engineering software service interface, for example, a particular CORBA (URL: http://www.omg.org) object.

The conceptual framework for mapping product model instances to software service interfaces adapts heuristic classification, described by Clancey (1985). He noted that successful expert systems abstract domain data into a set of concepts that map to a hierarchy of pre-enumerated solutions by nonhierarchical, heuristic match. The solutions are

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

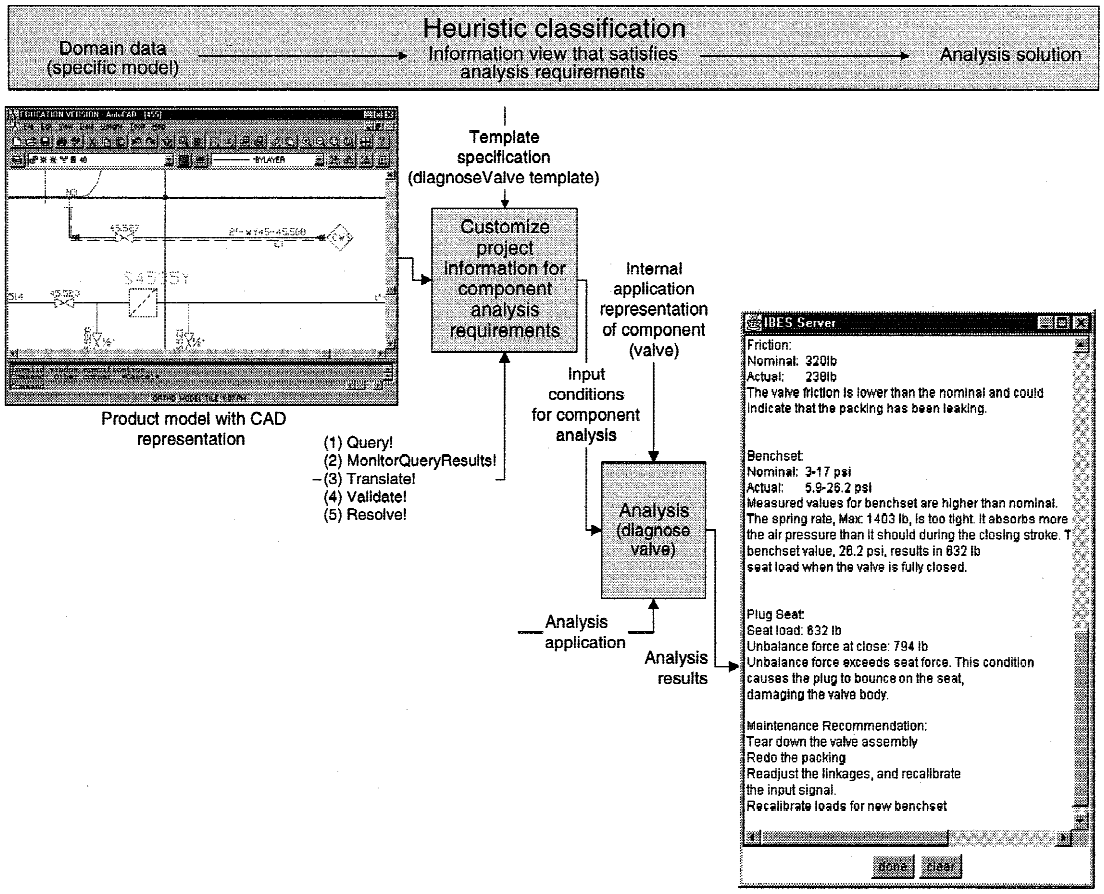


Fig. 4. IBES process to develop data input for analysis (IDEF0).

then refined. IBES applies similar principles for mapping product model instances to analysis application input conditions.

Figure 4 illustrates how we use data extraction and translation concepts from database integration research (Chaudhri et al., 1998; Genesereth et al., 1997; Maluf & Wiederhold, 1997) to abstract source information from generic data models to information views that satisfy the input requirements for component analysis. In an abstraction framework to support knowledge composition, Maluf and Wiederhold define articulation axioms (rules) that translate domain databases to an object-oriented reference model. Based on a dictionary of terms, they then unify knowledge in the reference model for interoperation. This step occurs through articulation rules that unify terms with respect to an explicit representation of their definition context in the dictionary. The knowledge abstraction process produces a common workspace for knowledge manipulation and use according to an ontology algebra (Maluf & Wiederhold, 1997).

IBES applies query and translation (1 and 3) of domain data in a standard model (data extraction) to views in a reference model (PLIB). It checks for incomplete information resulting from query (2), and validates views against the template specification of input conditions (4). Extraction enables the use of separate analysis applications implemented as an Internet service.

For example, the valve-sizing application requires a valve *Inlet_Pressure* value (see Fig. 6) for the minimum, maximum, and upset design cases of the valve. The *pressure* attribute of a *stream_design_case* instance in the source STEP model provides this information, see Figure 2. The *Inlet_Pressure* attribute of the *cvSizingTemplate* constrains the external query (1) of project information through an *interpretationMap* object (see Fig. 5) that relates the attribute to a query expression for the AP227 schema. For *Inlet_Pressure*, the query searches in the AP227 model for a *stream_design_case* instance that relates to the pipe component upstream of the valve.

In summary, IBES reasoning to extract product model data for component analysis includes the following steps:

1. Select a predefined query given system identification of the product model schema and the user's choice of a component engineering analysis;

2. Generate goal conditions for each data element of a query expression to monitor and report incomplete information that results from a query;

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

Table 2. *Shared and distinct input conditions for the test cases after extraction from source models*

Preliminary Valve-sizing Conditions	Shared Input Conditions	Valve Diagnosis Input Conditions
Logical identifier	✓	Logical identifier
Valve type (class)	✓	Valve type (class)
Inlet pressure at valve (maximum, minimum, and upset controlled flow)		Inlet pressure, fully open valve
Outlet pressure at valve (maximum, minimum, and upset controlled flow)		Inlet pressure, fully closed valve
Stream temperature		Outlet pressure, fully open valve
Steam flow rate (maximum, minimum, and upset controlled flow)		Outlet pressure, fully closed valve
Stream material		Valve shutoff class
Stream phase (non turbulent, turbulent, gas, and vapor)		Valve port diameter
Nominal valve inlet diameter		Nominal actuator travel
Nominal valve outlet diameter		Actual actuator travel
Pipe class		Nominal bench set
Pipe material		Actual bench set
		Actual actuator pressure
		Nominal actuator pressure
		Actuator stroke period
		Actual actuator stem friction
		Actual actuator seat load
		Nominal actuator seat load
		Actuator stem diameter
		Actuator diaphragm effective area
		Nominal actuator stem friction

3. Translate project information returned by query to the reference model vocabulary;
4. Validate (see also Table 3) data to match translated information against the template specification of analysis input conditions (Fig. 6);
5. User interaction to resolve ambiguities found during validation.

We used the PLIB resources to define the component libraries and the IBES reference model. PLIB served as a natural reference notation for our modeling purposes because, by definition, PLIB models abstract the representation of components and component functional models from a product model that represents such information in the context of a facility design.

Figure 5 describes the primary PLIB schema fragments used for our research and shows additions (gray) that we added to implement IBES. The object definitions in Figure 5 enable understanding of the instance models in Figure 6. Note the minor additions (template, view, and viewComponentClass definition) that support view extraction from product model data for bi-directional communication with component analysis applications.

In the PLIB architecture the definition of any semantic concept derives from the root class *dictionaryElement*, for example, a component, a component property, or a component functional model. All *dictionaryElements* are fundamentally defined by an absolute identifier (attribute of a basic semantic unit object), a prescribed list of properties, and relations with other objects (is_a, is_part_of, contains, is_case_of, etc.). We added template and view classes as a specialization of the *dictionaryElement* topology. Furthermore, PLIB supported our requirement for the definition of views from generic models by describing component features, functions, and views through three classification hierarchies:

- General model representation that defines component features and materials by product family;
- Functional model representation that defines the context parameters for placement of a component in a design;
- Views that add system attributes to represent a general and/or functional model in an application context, for example, a CAD 3D coordinate system.

6.1. Case reference models

With our semantic extensions, the PLIB representation syntax and standard set of entities provided a common workspace for IBES reasoning to monitor extracted data for completeness and validate it for correctness against component analysis input conditions specified by templates. This reasoning enables bi-directional communication and a web-based link between separated product model and analysis nodes.

Figure 6 is an instance diagram that describes a template and the template property, *Inlet pressure*, after IBES extracted information from the source model and generated

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

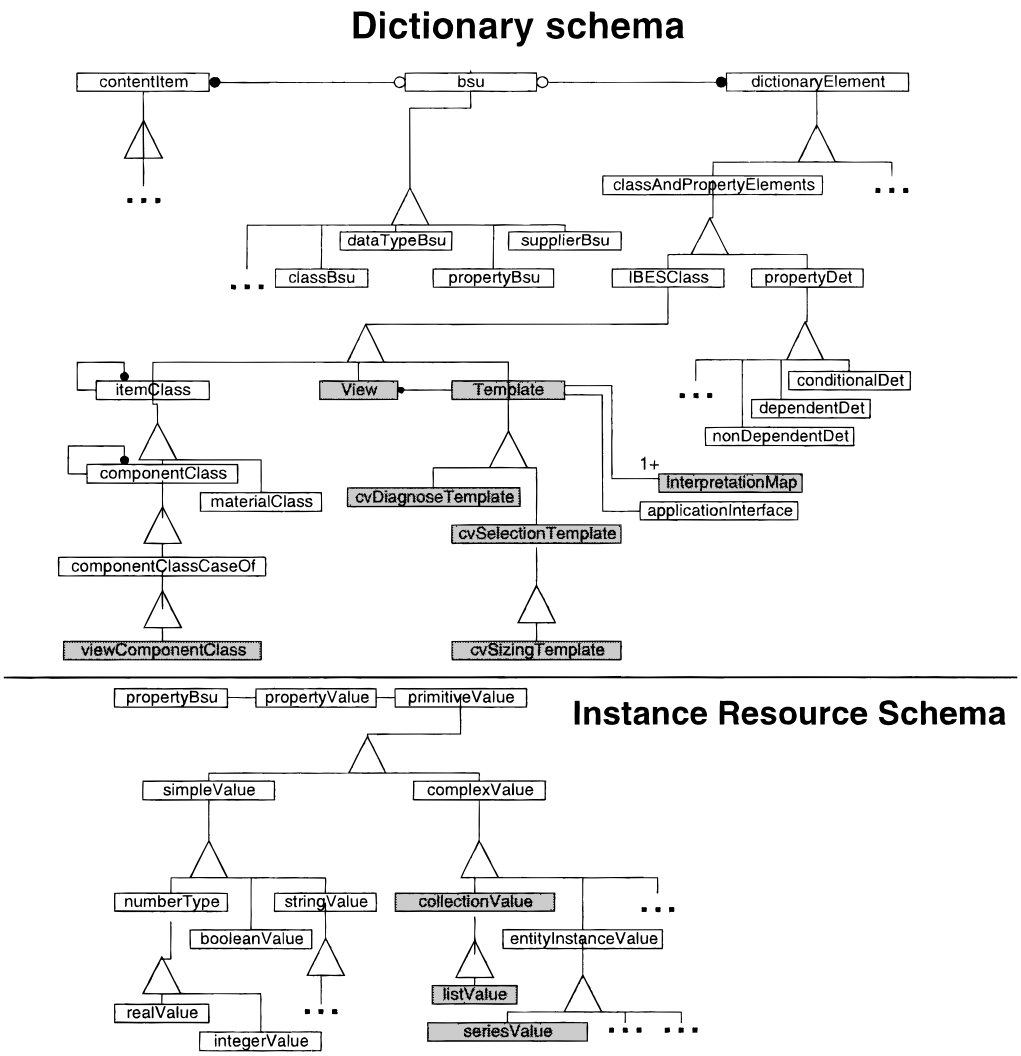


Fig. 5. PLIB schema fragments (OMT notation) and IBES extensions (gray).

views of information for input to component analysis. The elements with a shaded background represent instances of the control valve-sizing template, *cvSizingTemplate*, and the definition entities for the *Inlet_Pressure* property. The elements with a white background represent project information retrieved by query and translated into the reference model syntax. The *propertyValue* references a *realValue* that contains the real number, 20.0 for *Inlet_Pressure*. The consistent representation for the template definition and view instances provides a common workspace for extraction reasoning.

Table 3 describes in pseudo-code three of the production rules that validate information retrieved by query. The actual product models, IBES model, and production rules are written using Rete++, an inference engine and Windows 95™ C++ application programming interface that extends the 'C' language Integrated Production System (CLIPS). The pseudo-code assumes that variables (preceded by a ?) are assigned values through pattern matching with other clauses through *unification* (Stefik, 1995). CLIPS uses unification to assign values to variables.

The rules (for example Table 3) pattern match information represented by a view (left side of Fig. 6) with the input condition specifications of a template (right side of Fig. 6). Validation for assembly objects, such as the valve assembly in Case 2, requires match of a more complicated view and template representation.

The rules reference control objects, *container_match_controller* and *property_controller*, defined by IBES to support validation reasoning. IBES generates these control objects at run time to monitor the match status of views, view properties, and view property characteristics that are generated by query. A modification or change in the value of a property characteristic changes an associated control object via dependency rules. The state of the control objects, whether a control is satisfied, not satisfied, or has an "unknown" state, governs the activation of the validation rules.

Validation is complete when all the properties of a view match a template's input condition specification. For example, activation of the validation rule set for preliminary valve sizing begins with the goal to invoke the application. A ser-

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

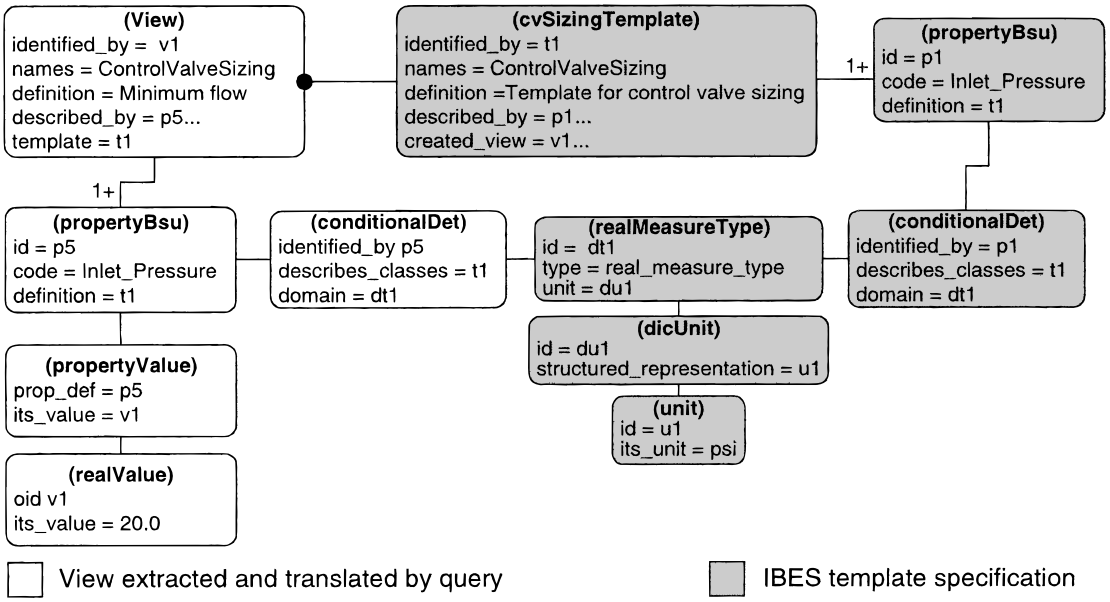


Fig. 6. Template definition and extracted valve for *Inlet Pressure*, research Case 1 (OMT notation).

vice match controller monitors the satisfaction of this goal by indicating whether all of the view instances generated by query match the specification furnished by *cvSizingTemplate*. A view, for example v1 (Fig. 6), is validated when all its properties, for example *Inlet_Pressure*, match their template specifications. The rule, *View Match* (Table 3), checks that all properties of a view have been validated. As the rule indicates, a *container_match_controller* monitors the satisfaction of this subgoal for each view. Similarly, a property is validated when all its characteristic values match their specification. The second rule in Table 3, *All Property Characteristics Match*, checks for this state of a property. The third rule in Table 3, *Code Match*, is an example of a property characteristic check. Specifically it validates the code characteristic. Other rules check for domain and unit representation characteristics. For example, the value real_measure_type and psi for the *domain* and *unit* characteristics of *Inlet_Pressure* must match their template specification. The rule shows how a *property_controller* monitors the satisfaction of this subgoal for each property instance.

Because template and view representations abstract information from an implementation context, the validation algorithm is general for any view–template relation. Property characteristic validation assures that information obtained from the product model is relevant and sufficient for invoking the analysis application. Additionally, the state-based nature of the system makes the validation process dynamic. Finally, the pattern matching validation algorithm facilitates easy identification of incomplete and mismatched data and the invocation of the application according to the successful match of the underlying data.

7. DISCUSSION

To support life cycle engineering, we looked at two engineering processes that are typical for different phases of the facility life cycle and the standard product models that are designed to enable such processes. The models for each process required modest extension: the definition of standardized queries, and structural modification to add semantics and reasoning that support bi-directional communication for component analysis. IBES reasoning for information extraction and integration with component analysis was consistent for the research cases. We conclude that it is possible to design a general set of methods that integrate routine component analysis with project information models, and support Web automation for AEC engineering services.

We were pleased that only modest extensions to product data representations were necessary to enable these representative engineering analyses. However, the prospect of many developers of engineering services requiring modest changes suggests that the current vision of standard product models will be difficult to realize in practice.

In the Background section, we argued that engineering analysis requires a suitably detailed representation of the form, function, and behavior of the modeled product. We observed that, for the two test cases described in this paper, little form representation was shared, and no function and no behavior were shared. These distinct applications shared only object identifier and name, evidence that large and complex information models did not contribute in this case to integration of distinct analysis applications. One additional realistic example of engineering analyses that share behavior is bi-directional

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

Table 3. *Sample validation rules (pseudo code)*

Rule	Comment
VIEW MATCH	
FOR container_match_controller ?c WHERE ?c.spec_object == ?id AND ?c.container == ?vid AND ?c.spec_prop_count == ?x AND ?c.spec_prop_count /= 0 AND ?c.container_prop_match == ?x AND ?c.container_match == FALSE AND Template ?t WHERE ?t.identified_by == ?id AND View ?v WHERE ?v.identified_by == ?vid THEN ?c.container_match = TRUE	The view match rule checks that all properties of a view match the template specification.
ALL PROPERTY CHARACTERISTICS MATCH	
FOR container_match_controller ?c WHERE ?c.spec_object == ?id AND ?c.container == ?vid AND ?c.spec_prop_count == ?count AND ?c.container_prop_match == ?m AND ?c.matched_properties == $?props AND Template ?t WHERE ?t.identified_by == ?id AND ?t.identified_by /= nul AND IBESClass ?ic WHERE ?ic.identified_by == ?vid AND ?ic.described_by == $?list AND FIND ?vprop in $?list AND property_controller ?pc WHERE … AND ?p.is_matched_property == FALSE AND NOT MEMBER(?vprop $?props) THEN If (OR (AND (?unit_satisfied == TRUE) (?applicable == TRUE))(?applicable == FALSE) THEN int ?x= (?matches + 1) ?c.container_prop_match == ?x APPEND(?c.matched_properties ?vprop) ?pc.is_matched_property == TRUE	A property matches when all of its characteristics match (code, domain and unit representation if applicable). The “all property characteristics match” rule checks that all characteristics of a view property match a corresponding template property.
CODE MATCH	
For property_controller ?pc WHERE ?pc.spec_property == ?prop AND ?pc.container_property == ?vprop AND ?pc.code_is_satisfied == ?cis AND ?cis == UNKNOWN AND property_bsu ?bsu1 WHERE ?pbsu1.code == ?code AND property_bsu ?bsu2 WHERE ?pbsu2.code == ?code AND IBESclass ?c WHERE ?c.described_by ==$?list AND ?c.definition == ?definition AND MEMBER(?vprop $?list) THEN ?pc.code_is_satisfied = TRUE	The code match rule checks the code characteristic of a property. Similar rules exist to check domain and unit representation if applicable. In the event a characteristic is not extracted from the product model, or does not match, the match status remains “UNKNOWN” and is flagged for user modification.

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press

reference to production rates for construction cost and schedule applications. In our research, we have not identified other realistic examples of significant sharing of behavior that are well founded in engineering theory. Our findings on minimal information sharing corroborate preliminary evidence from previous CIFE research that describe shared geometry for analysis (Kunz et al., 1996) but little other shared information among applications. Given the high overhead and difficulty of developing models that represent the union of all information requirements, our results lead us to question the assumption that such models will be effective or contribute toward integrated engineering.

AEC product models can have two different integration objectives (Koskela, 1995). One objective is to support data sharing among like applications—for example, two CAD applications or two valve-sizing applications. In this case, the shared model specification can appropriately include the different behaviors that the applications support, requiring enough description of forms and design intent to enable clear description of behaviors and the context in which they are found. Another possible objective is to support data sharing among unlike applications—for example, for a CAD application to provide information for a valve-sizing application. In this case, the shared model specification needs only the modest amount of form that must be shared between intelligent applications, with little requirement to represent design intent or extensive behaviors. This latter goal is more easily implemented. In the spirit that "simplicity wins," it may also be tractable in practice.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Integrated Facility Engineering at Stanford University and the National Institute of Standards and Technology, Computer Integrated Construction group.

REFERENCES

Aalami, F.B., Fischer, M., & Kunz, J.C. (1998). *AEC 4D Production Model: Definition and automated generation.* CIFE Working Paper, Center for Integrated Facilities Engineering, Stanford University, Stanford, CA.

Amor, R., Hosking, J., Groves, L., & Donn, M. (1993). Design tool integration: Model flexibility for the building profession. *Proc. Building Systems Automation—Integration.*

Bjørk, B.C. (1989). Basic structure of a proposed building product model. *Computer-Aided Design 21(2)*, 71–78.

Chaudhri, V.K., Farquitar, A., Fikes, R., Karp, P.D., & Rice, J.P. (1998). OKBC: A programmatic foundation for knowledge base interoperability. *Proceedings of AAAI-98*. Madison, WI, July 1998.

Clancey, W.J. (1985). Heuristic classification. In *Artificial Intelligence 27*, pp. 289–350. Elsevier Publishers B.V., North-Holland.

Clayton, M.J., Fruchter, R., Krawinkler, H., & Teicholz, P. (1994). Interpretation objects for multidisciplinary design. In *Artificial Intelligence in Design, '94*, (Gero, J.S. and Sudweeks, F., Eds.), Kluwer Academic Publishers, Dordrecht, the Netherlands.

Driskell, L. (1983). *Control-valve selection and sizing.* © Instrument Society of America. Publishers Creative Services, Inc., ISBN 0-87664-628-3.

Eastman, C. & Sheng Jeng, T. (1998). A database supporting evolutionary product model development for design. In *Automation in Construction.* Elsevier Science B.V., North Holland.

Fitzgerald, W.V. (1990). *Automated control valve trouble-shooting: The key to optimum valve performance.* In Instrumentation Society of America paper #90-142, by courtesy of Fisher Controls International, Inc.

Genesereth, M.R., Keller, A.M., & Duschka, O. (1997). Informaster: An information integration system. *Proc. 1997 ACM SIGMOD Conf.*

Instrument Society of America, ISA-S75.01 (1991). Standards and Recommended Practices for Instrumentation and Control, 11th ed. Research Triangle Park, North Carolina.

International Standards Organization, Technical Committee ISO 184, ISO CD 10303-221 (1997). Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages—ISO 10303 Product data representation and exchange—Part 221 Functional data and their schematic representation for process plant.

International Standards Organization, Technical Committee ISO 184, ISO CD 10303-227 (1994). Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages—ISO 10303 Product data representation and exchange—Part 227: Plant Spatial Configuration.

International Standards Organization, Technical Committee ISO 184, ISO CD 13584-10 (1995). Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages—ISO 13584, Parts Library—Part 10: Conceptual descriptions: Conceptual model of parts library.

Koskela, L. (1995). On foundations of construction process modeling. *Proceedings of the CIB Workshop on Computers and Information in Construction, CIB W78*, 503–514.

Kunz, J.C., Jin, Y., Levitt, R.E., Lin, S.D., & Teigholz, P. (1996). The intelligent real-time maintenance management (IRTMM) system: Support for integrated value-based maintenance planning. *IEEE Expert, 11(4)*, 35–44.

Maluf, D.A. & Wiederhold, G. (1997). Abstraction of representation for interoperation. Foundations of intelligent systems. In *Artificial Intelligence, Subseries of the Lecture Notes in Computer Science* (Zbigniew, R. and Skowron, A., Eds.) Vol. 1315, Springer Verlag, New York.

Stefik, M. (1995). Introduction to knowledge systems. Morgan Kaufmann Publishers, Inc. San Francisco, CA.

URL: http://www.interoperability.com/

URL: http://www.mel.nist.gov/sc5/soap/

URL: http://www.omg.org

J. Andrew Arnold recently completed a Ph.D. and worked as a research associate at the Center for Integrated Facilities Engineering (CIFE), Stanford University. In May 2000 he joined the CMD Group as director of e-Commerce Content. His research interests include product and process modeling to support key business activities in Architecture, Engineering, and Construction. He also has interest in database integration and knowledge composition methods for semantic interoperation of information from multiple knowledge sources. Prior to his graduate studies, Andrew studied Architecture and Environmental Design at the College of Environmental Design, U.C. Berkeley.

John C. Kunz is a Senior Research Engineer at the Center for Integrated Facility Engineering (CIFE) in the Department of Civil Engineering at Stanford University. His research interests include: non-numeric (Symbolic) Modeling of Engineering Products and Processes, education, training and technology-tranfer. Dr. Kunz studied Engineering Science at Dartmouth College and the Thayer School of Engineering, and Computer Science at UCLA and Stanford. He has published in areas of physiological modeling, computers in medicine, artificial intelligence, manufacturing, and civil engineering.

https://doi.org/10.1017/S0890060400142039 Published online by Cambridge University Press