Requirements on Information Technology for Product Lifecycle Management

Peter Denno
National Institute of Standards and Technology,
Gaithersburg, Maryland, USA
peter.denno@nist.gov

Thomas Thurman
Rockwell Collins,
Cedar Rapids, Iowa, USA
trthurman@rockwellcollins.com

ABSTRACT

Good decision-making is founded on good information. Information technology supporting product lifecycle management ought to provide a high degree of information cohesion and traceability – knowledge of the interrelations among data, and basis for belief. Providing cohesion and traceability is made difficult by differences in viewpoint and ontology employed by the various disciplines and organizations involved in the product lifecycle. This paper describes an analysis of cohesion and tracability into its constitutents properties. The paper suggests that *process-aware integration schema* can improve the cohesion and traceability among product data.

KEYWORDS: Product Lifecycle Management, Information Quality, Product Design, Integration Schema, Information Modeling.

1 Introduction

Product lifecycle management (PLM) concerns the management of *lifecycle data* - data associated with an actual instance of a product that records an account of its development, deployment, operation, maintenance, and disposal. Interest in PLM has brought focus and renewed energy to the general problem of managing technical product data. However, these challenges have existed for several years, and for the most part, PLM is a perspective on the general problem rather than a specific solution. This paper aligns with the findings of a workshop on product data management hosted by NIST [Workshop-2]. In that workshop, the notions of information *cohesion* and *traceability* were found to be crucial to improved product data management.

This paper considers the nature of cohesion and traceability and describes its constituents. The constituents reflect requirements on PLM systems. The paper describes how the constituents have been addressed elsewhere by various technologies. However, what is needed is a comprehensive solution, not one focused on individual constituents in isolation. We therefore present a general architecture for a system supporting PLM based on a "process-aware" integration schema.

Section 2 of the paper enumerates requirements for supporting cohesion and traceability of information used in the product lifecycle. Section 3 relates these requirements to particular technologies, including information modeling and integration schema. Section 4 introduces the idea of a process-aware integration schema and illustrates, through an example, its role in providing the constituents of cohesion and traceability. The paper is concluded in Section 5.

2 Cohesion and Traceability

Cohesion is knowledge of the interrelationships that exist among data. *Traceability* is knowledge of the origin or basis for belief in certain data. Problems of cohesion and traceability are particularly severe in engineering environments *vis a vis* other domains (*e.g.* financial) because of the nature of engineering products. For example, (1) the relationships among product requirements, structure, function and, behavior are complex; and, (2) regulations and quality certifications may constrain designs and require data on in-service products.

A principal conclusion coming out of the NIST workshop [Workshop-2] was that current data management practices do not provide sufficient support of information cohesion and traceability. Cohesion and traceability, however, are complex and abstract properties when viewed as attributes of an information system. Information technology does not address these goals directly, rather certain other qualities help to support these goals.

We distinguish the following nine *constituent properties of cohesion and traceability*:

- **associativity across views** : We distinguish two kinds: (1) A *conceptual gap* is the absence of the knowledge that two conceptualization can be used for the same purpose; and, (2) An *associativity gap* [Peak-2] is the absence of the knowledge that two references refer to the same thing.

- **effectivity** : a qualification, defined in terms of the configuration of a product, its production date or serial number, that determines whether the data applies to a particular instance of the product. "Instance of the product" here might be a conceptualization, such as a part family. Effectivity is a *modality,* a qualification on the truth of an assertion.

- **authority** : the power that data or an assertion has due to an approval it is granted or an estimate of its maturity. Approvals and estimates of maturity are ultimately underwritten by a role in the organization.

- **origin in requirements** : relationships that indicates how the data are related to requirements (*e.g.* design rationale, trace relationships). This is a systems engineering sense of traceability.

- **origin in process** : relationships that indicate how (through what process) the data were produced, validated, authorized for use, or constrained. This is an engineering management sense of traceability.

- **origin in media** : relationships that indicate where the data reside. “media” might not be as concrete as “on the disk of computer xyz” but rather “in the CAD repository of designs approved for manufacture.” This is an information technology sense of traceability.

- **origin in other belief** : relationships that indicate the logical support for the data (*e.g.* validation testing results, an idealization described in mathematics, component characteristics from the supplier). The logical support of a belief may change over time. The assumptions on which it rests may no longer be true, or may be found to have been false all along. This is a logical sense of traceability.

- **logical consistency** : this includes *type awareness*, *interpretation constraints* and *well-formedness conditions*. Type awareness concerns knowledge of the type of thing to which the data refers. Type reference data may include units of measure. Interpretation constraints are assertions about what roles a particular type can serve. For example, a resistor can be part of an electrical circuit. Well-formedness conditions constrain the structure of the data to be consistent with the system of expression (syntax, domain of values *etc.*) in which it is defined.

- **measurement conditions** : (if the datum is a measured or computed value) reference to the process by which it was measured or computed, degree of confidence, or a statement of numerical accuracy.

These constituent properties provide a context to the data that the system manages. Together they determine the confidence one places in the data and thus its power to affect the course of action. The constituents describe an epistemological basis of product lifecycle data.

3 Constituents and Technologies

Each constituent reflects a set of requirements onto the PLM system. In some instances, there are obvious relationships between those requirements and certain technologies. For example, mapping languages [Express-X], engineering frameworks [Peak-2], and integration schema [ISO15926-2] address associativity gaps. Nonetheless, possessing a solution in an isolated technology is usually not sufficient. The constituents interrelate in ways that make it impossible for the total solution to be successful when each is addressed in isolation. More optimistically, technologies can be reused to address multiple constituents. For these reasons, the challenges of cohesion and traceability require a comprehensive architecture, not piecemeal solutions. However, before we describe such an architecture, we briefly overview some of the strongest links between particular constituents and technologies.

3.1 Associativity across views: integration schema

The various engineering disciplines use differing tools and models in their work. When communicating across disciplines, information must be mapped from the viewpoint and models of one discipline to the viewpoint and models of another. An *associativity gap* is the absence of the knowledge that two references made from different viewpoints refer to the same thing. [Peak-2] Often, human intervention is required to bridge associativity gaps. It has been estimated that about one million *fine-grained associativity gaps* are bridged in the structural analysis of an airframe [Peak-2].

Associativity gaps are distinguished from conceptual gaps as different usages of the term "semantics." The term is often used ambiguously. By semantics, we mean either how a thing is *referenced* or the *sense* of the thing [Frege], [Quine]. Frege's oft-cited example is that of the "morning star" and the "evening star." The morning star appears in the morning in the east. The evening star appears in the evening in the west. However, both terms often refer to the same celestial object, Venus. Reference integrity may be necessary but sense consistency (even one reflecting an incomplete understanding of the domain) is often sufficient. The context of the problem determines what model is needed. In the case of the morning and evening stars, ancient navigators did not need to be aware of the equivalence among references. Sense consistency is sufficient for their purposes.

Generally speaking, associativity gaps occur in data, and conceptual gaps occur in schema, or across multiple schema. For example, a CAD model might define the tolerance of a geometric feature. A tolerance stack-up analysis might require the tolerance of that same feature. If we do not know that the feature in the tolerance analysis is the same feature as in the CAD model, then it is impossible to make use of the tolerance values as defined in CAD model. This is an associativity gap. It is bridged by explicitly equating references among things. Bridging associativity gaps makes available additional information about the thing referenced. On the other hand, a planning system may not be able to obtain the duration of a task because it has been provided with a start time and a stop time, but not the fact that duration is stop time minus start time. This is a conceptual gap. It is bridged by defining logical relationships among elements of the schemas governing the data. Bridging conceptual gaps makes existing data governed by the schemas more useful, without explicitly referencing any of that data.

When a conceptual gap concerning *identity conditions* [Guarino] is resolved, it may be possible to resolve associativity gaps among the individuals identified by those conditions. An *identity condition* is a property expressed in the data by which one can distinguish individuals. Thus *employee-number* would be an identity condition for employees in some database if it uniquely identifies employees. Identity conditions may differ across viewpoints. In the CAD / tolerance model example above, geometric position on the part is an identity condition for geometric features. However, the CAD model and the tolerance model might use differing coordinate systems to identify the feature.[1] If these differences can be resolved, the solution to the associativity gap can be expressed as a binary predicated (e.g. *same-position*) with tuples naming features in the two viewpoints.

Conceptual gaps that concern identity conditions are resolved by equating identity conditions across viewpoints. Conceptual gaps that do not concern identity conditions, such as illustrated by the planning system example above, may require domain knowledge and use of a logical language to state the conceptual relationship (*e.g.* that duration is stop time minus start time).

Integration schema [ISO15926-2], [EIA-927] are schema designed to accommodate and interrelate multiple discipline-centric viewpoints of their subject matter. These discipline-centric viewpoints are represented by systems and databases of their own. Integration schema address both associativity gaps and conceptual gaps.

Typically, integration schema contain provisions to accommodate differences in conceptual scope and conceptual factoring (*i.e.* choice of concepts covering the domain). However, some schema achieve the goal of accommodating multiple viewpoints without these special provisions because the viewpoints within their scope are known in advance. The STEP electromechanical application protocol, AP210 [ISO10303-210], for example, is an integration schema of this sort. It provides usage and design views, including a functional design view, and the encapsulation of simulation views. The difference between AP210 and things nominally referred to as integration schema derives from the anticipation of scope and usages in the former. Because AP210 concerns a cohesive subject matter known *a priori* (electromechanical products), most of the problems of incompatible conceptual scope and conceptual factoring were anticipated during schema design and have been eliminated.

Typically, integration schema designers do not have the luxury of knowing the embodied viewpoints in advance. Instead, the integration schema may serve as a meta-model to those viewpoints it concerns. The meta-model allows the integration schema to "late bind" semantics from those viewpoints by embodying, as data, aspects of the discipline-specific schema. The meta-model helps to identify relationships among concepts in the various viewpoints. This is not a role commonly associated with meta-models (*e.g.* the UML meta-model [UML]) and accordingly, integration-schema-as-meta-model provide

1 This illustrates the added challenge of resolving associativity gaps among engineering data. In the employee database the identity condition is a key in a relational table, and is documented in the schema as such. In the engineering data, geometric transformation may be necessary and issues of numerical precision must be considered.

an unusually elaborate high-level conceptual model[2] when compared with these other meta-models.

In its role as a meta-model to discipline-specific tool schema, the integration schema serves to eliminate conceptual gaps.

Databases built on the integration schema do not only embody discipline-specific viewpoint schema, but also instance data produced from systems supporting those viewpoints. It may not be possible (nor necessary) to embody all the information from discipline-centric viewpoints. Instead, the database may record only information that improves cohesion and traceability. The database may also note the creator and creation time of entries in the originating discipline-centric systems and databases. [ISO15926-2]

In its role as a repository for data created by discipline-specific tools, the integration schema serves to eliminate associativity gaps.

3.2 Logical consistency: information modeling

The constituent *logical consistency* concerns *type awareness*, *interpretation constraints* and *well-formedness conditions* on information. These requirements are addressed by information modeling languages, including conceptual modeling languages, ontologies, and related methods.

Information modeling languages serve prescriptive and descriptive purposes. In a prescriptive role, the language is used to define schemas which “govern” data; the focus here is on the well-formedness of the data as a whole. Typically that “whole” is the unit of exchange among software tools. In a descriptive role, the language serves to distinguish types within the population, and defines relations on those types; the focus here is on providing a viewpoint on populations that range without regard to fitness to a particular application. Descriptive provisions primarily address the population whereas prescriptive provisions primarily addresses data about the population. By design, modeling languages vary with respect to how well they address prescriptive and descriptive needs. EXPRESS [ISO10303-11], for example, is predominantly used in a prescriptive role, accordingly it provides an abstract data type language by which it specifies well-formedness conditions on a population of data entities built from primitive types. Ontologies, on the other hand, typically serve a descriptive role and concern the population directly.

The sense of well-formedness, as it relates to the exchange of data, varies among languages. The abstract data type language of EXPRESS explicitly specifies what

2 The “high-level conceptual model” elaborates upon concepts in the neighborhood of “thing.” The EPISTLE core model version 4.2, for example, distinguishes things that can exist (“individuals”) from conceptual things; “single individuals” from “plural individuals” *etc*. This model is a “conceptual meta-model,” unlike the “storage meta-model” that is the UML meta-model.

properties are to be included in the exchange of EXPRESS entity instances (those properties that are the mapped attributes of the entity type). This is one purpose of the EXPRESS notion of well-formedness. XML Schema (though not an information modeling language *per se*) can specify similar constraints on the form of XML files. On the other hand, the Web Ontology Language (OWL) [OWL] makes no such stipulation. OWL does not specify what properties of the instance are to be included in an exchange. Separating concept modeling issues from exchange issues (and dealing with exchange issues where they arise using XML Schema) is a common approach in more recent modeling languages.

In addition to these considerations, modeling languages provide various solutions to the following issues (among others):

- **associations/attributes** : Some languages (*e.g.* entity-relationship languages) do not distinguish attributes from associations (Attributes are relationships between an entity type and a value type. Associations are relationships among entity types.)

- **open/closed world assumption** : A closed-world model entails that if some assertion cannot be shown to hold, then the negation of that assertion hold. (*e.g.* If it is not the case that *p(x)* then *NOT p(x)*.)

- **primitive/defined distinction** : Some languages, particularly description logics, explicitly distinguish "defined" concepts from "primitive" concepts. Defined concepts are concept for which necessary and sufficient conditions are provided. Primitive concepts are concepts for which conditions stated may be necessary but are not sufficient. A *conservative definition* does not introduce a new primitive concept.

- **meta content** : Some languages embody definitions of their own modeling primitives. In the case of Suggested Upper Merged Ontology (SUMO), [SUMO-1], [SUMO-2]for example, some of the representational machinery is defined using itself. (KIF [KIF]is also used in SUMO).

- **class/instances distinction** : Some languages bridge the distinction between classes and instances; in these it is possible to treat a class as a kind of instance.

- **information structure** : Some languages, particularly information and object modeling languages, allow modelers to define "entity types" as structures built around value types (strings, integers *etc.*).

- **data types** : Some languages identify certain value spaces (*e.g.* strings, numbers, point in time), as primitive building blocks of other types. They may define an encoding scheme on those types for use in data exchange.

- **quantifiers** : Some languages, such as those based on first order logic, have explicit quantifiers (*e.g.* “for all” and “there exists”). These are used to express the quantity of individuals in a population (zero, at least one, all) that satisfy some condition.

N.B. Where a language falls with respect to these issues cannot in itself be judged as an advantage nor disadvantage. What is an advantage in one use of the language may be an obstacle in another. For example, it is probably wrong for a conceptual modeling language to distinguish attributes from associations. Conceptual modeling languages, their purpose being to distinguish types of individuals in some population and draw relationships among those types, should not make commitments with respect to attributes and associations. [Halpin2]

3.3 Origin in process: process specification

The activities of the product lifecycle occur in processes (patterns of activity). Documented processes may be required by regulation and for certification, (*e.g.* ISO 9000 certification, [ISO9000]). Informal, undocumented processes occur in all manufacturing activity. The explicit specification of processes can help management in decision making (*e.g.* process re-engineering) and provide input to automation (*e.g.* workflow systems).

The Process Specification Language (PSL) is a formal, model-theoretic ontology for process specification. PSL consists of core theories and extensions. This modular approach, and a method of axiomatizing concepts from other process specification languages [Gruninger], makes PSL suitable for the integration of information from other software tools. [Cheng]

The definitional extensions of PSL contain concepts needed for reasoning about industrial processes including conditional, triggered and iterated activities, duration-based constraints, and reusable, consumable, renewable, and deteriorating resources. [Gruninger]

For our application, a particularly important attribute of PSL relative to other process specification method is its ability integrate with an ontology. Process definitions in PSL extension theories provide the notions of precondition and effects, which are used to describe changes of state caused by execution of an activity occurrence of the process. These changes of state concern domain entities (in our case, product components, machines *etc.*). Both PSL and SUMO are defined in KIF. If the two were used together, SUMO-based concepts could be referenced as preconditions and effects.

4 Process-aware integration schema

Integration schema provide a tool for the integration of diverse information. Systems based around integration schema interact with discipline-specific application systems and their associated data. Integration schema serve to address associativity across views and logical consistency as described above. A central point of this paper is that cohesion and traceability is increased when integration schema are augmented with process knowledge. The process knowledge used by these schema describes business processes associated with the data of the application systems. To elaborate upon this, note that the constituent *origin in process* provides meta-data that specifies the process[3] that produced, validated, constrained the application of, or authorized the use of, the subject data. It is important to note here that (1) cohesion and traceability concerns not only the business processes that produce the data, but also those that validate, constrain, and authorize them; and, (2) processes “point outside themselves” to constituents. That is, documenting the activity occurrence leads to documenting authority, effectivity, measurement conditions, or origin in other belief – provided that the process ontology itself is detailed enough to assert that a consequence of executing the process is to affect these constituents of the subject data. This point is illustrated in *Figure 1*.

3 More accurately, using PSL terminology, it is not a process (pattern of activity) that is documented as meta-data, but a *complex activity occurrence*. Processes are concepual whereas activity occurrences exist at a certain time and place.

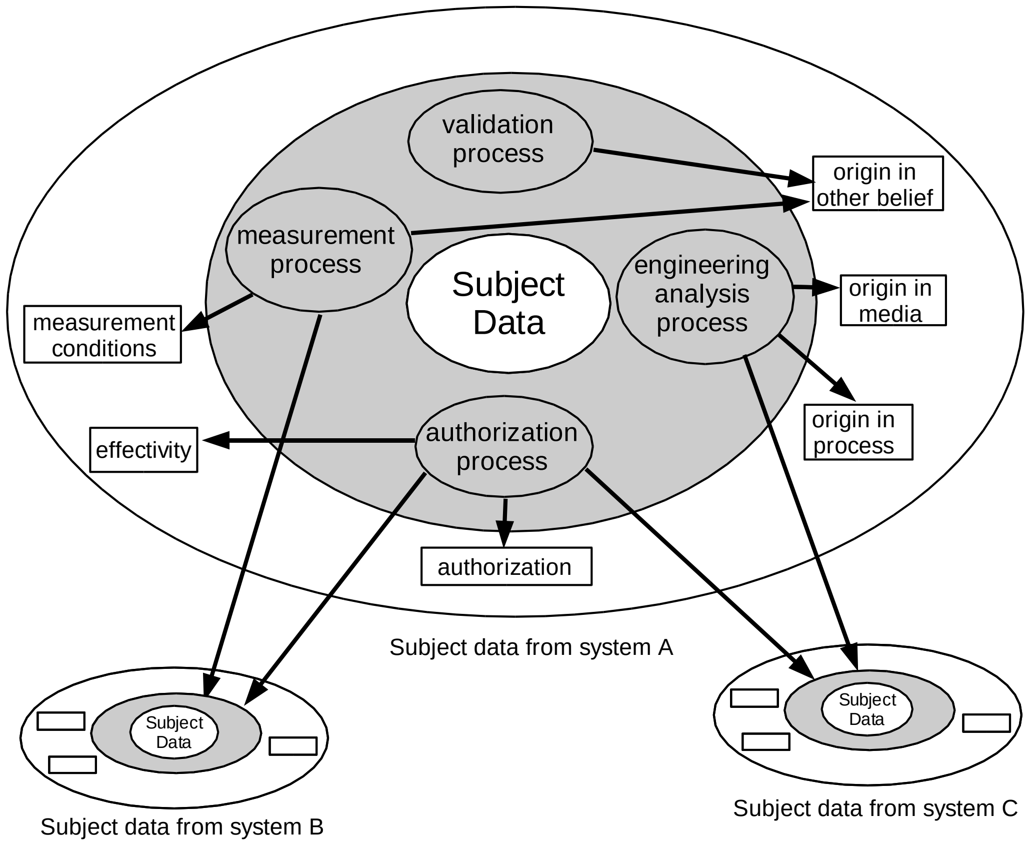

Figure 1: When activity occurrence meta-data is associated with the subject data, data from other application usage is related, and additional constituents can be realized.

Five constituents have strong links to process:

- Authority : information has the authority granted it by a formal, informal, or implicit approval process;
- Effectivity : an engineering change (EC) process determines whether data is effective with respect to a particular product instance or product configuration;
- Origin in other belief : the logical support for the data may be the result of a validation process, assumptions present in the idealization employed, or experience in the field;
- Origin in media : the presence of the data on a particular media (database, design repository etc.) is the result of process that uses the organization's information technology;
- Measurement conditions : in some situations, the conditions under which the measurement is made can be described by a process.

In order to illustrate how process specifications can help interrelate data that would otherwise remain isolated, we consider an example. Figure 2, is an IDEF0 [IDEF0] diagram depicting a design process for assembly on printed wiring boards (PWBs). In the process, components are placed on the PWB by use of a layout tool or mechanical CAD system. Design issues concerning the placement of a component include radio frequency

(RF) interference from other components (if the board contains RF circuitry) and thermal conditions during the soldering process. The placement of neighboring components, and the copper pads on the PWB to which they are soldered, contribute to the thermal mass of the region, and the thermal conditions the component experiences during soldering. The placement activity is followed by wire routing and then validation. Validation may include engineering simulations for thermal properties, producibility, and vibration. The design cycle of "place, route and validate" is repeated with incremental refinement of the placements until the engineering analyzes suggest that an acceptable design has been found.

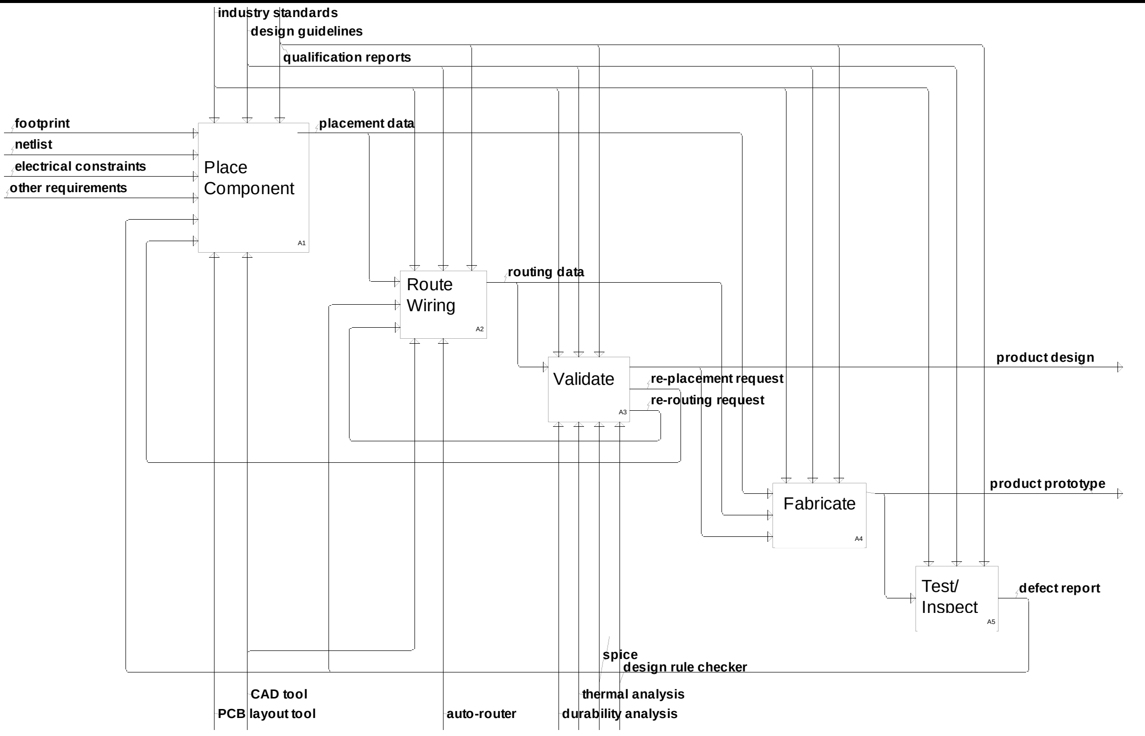

Figure 2 Circuit board design process

The "place, route, and validate" design cycle is embedded in an activity cycle which includes manufacture and inspection. Defects found on inspection are analyzed for causes, and results are integrated into the "lessons learned" of design guidelines. Of course, excessive defects can also lead to an engineering change order.

A common manufacturing defect occurring during the soldering process (part of *Fabricate* as depicted in *Figure* 2), is tombstoning of small components. *Tombstoning* is a phenomenon where a component is raised and detached from the PCB at one end while remaining bonded to the board at the opposite end. Tombstoning is due to uneven surface tension in the molten solder at opposing ends of the component. This can be the result of non-uniform cooling or heating, an excessive volume of solder paste, inaccurate placement of the component or paste, overly large pads, uneven thermal mass at opposing pads, and inappropriate solder paste alloy, among other causes. [Lim]

One role for process-aware integration schema in this environment is to bring together all the data that might bear on tombstoning, for example. At every intermediate point in the evolution of the information systems of the enterprise, a lack of access to potentially useful information results in less-than-optimal performance. For example, ideally the PCB component layout software would embody solder thermal dynamics knowledge that is intellectual property of the enterprise. But if the layout software is commercial off-the-shelf, it is difficult to see how this need could be realized in the layout software without benefiting competitors.

The following illustrate possible roles for a process-aware integration schema:

- **Identifying resources actually used :** The initial wire routing may be preformed with an automated tool. If that routing violates design rules, or does not pass validation, then subsequent re-routings may be performed manually with a CAD system. An activity occurrence can be registered in the integration schema for each routing design. Knowledge that the CAD tool was used in the activity occurrence (as opposed to the automated tool) indicates a deliberate act (*authority* constituent). Also, a history of such information may be used to improve the automated tool or estimate development cost.
- **Identifying process parameters** : The suppliers of components may recommend soldering process information, such as a *reflow profile* -- the ideal temperature gradient for soldering the component to the board. However, reflow ovens (used in soldering) by design must subject the entire board to single temperature gradient. Thus the reflow profile selected may be inappropriate for some components on the board. By defining reflow processes for each reflow profile, and associating components with the recommended process, a soldering activity occurrence that documents which process is invoked links component characteristic data to manufacturing process information. Here the integration schema serves to bind process information with component characteristics.
- **Identifying design parameters** : Likewise, suppliers of components may recommend the solder pad layout (geometric design of the mating surface on the PCB). Typically the PCB designer may choose a design that is widely applicable, but in problematic situations the designer may want to consider the recommendation of the component supplier.
- **Tracking upstream introduction of error :** the source of a problem detected during the validation or testing activities may be easier to trace through a hierarchical decomposition of activity occurrences and their process descriptions than it is by an analysis of transactions against a product data management system.

The examples above require fine grain process specification. The preconditions and postconditions of such processes may refer to concepts that would needed to be modeled. The cost of this modeling effort is significant, but is inevitable if relations across viewpoints are to be made. Further, in order to register information during actual processing, information mapping into the structures of the integration schema would also be required.

5 Conclusion

We described the requirements on PLM systems as being founded in constituents properties of data cohesion and traceability. We described a method to bind together, into a comprehensive process-oriented ontology, the product lifecycle data produced by the various software tools used by an enterprise. Presentation of this method is intended to illustrate aspects of the problem, not provide a solution of immediate value. Employing process knowledge in an integration schema increases the cohesion and traceability of the life cycle data.

Important obstacles to the implementation of these ideas exist, such as how applications can be wrapped, or output mapped, to produce entries in the integration schema. There is also the matter of making use of the added cohesion and traceability, which the method isolates into the integration schema. Note here that the method accounts for eight of the nine constituents, but *origin in requirements,* is not discussed in length. Our hope is that a system built around a process-aware integration schema can become the kernel of a systems engineering[4] tool, where systems engineering information (requirements, requirement allocations, property roll-ups, objective functions, *etc.*) can be layered over the integration schema database.

Assuming that these obstacles can be overcome, PSL and SUMO (with a manufacturing extension) seem particularly well suited to provide the process-ontology required by the integration schema. Depending on the industrial sector, AP210, the Core Product Model, or ISO 15926 appear to be best suited to provide other concepts in the integration schema.

The authors and their colleagues are developing AP210 and AP210 tools, the Core Product Model [Fenves], PSL and PSL tools, information mapping techniques and tools, [Express-X] and automated methods of integration [Denno]. This foundational work should lead to the ability to implement the central components of a process-aware integration schema.

6 Acknowledgments

The authors would like to thank all participants in the two NIST workshops (participant list at [Workshop-1]). Since that time, the following persons have provided stimulating discussion and critique that has certainly improved our thinking: Steve Fenves, Ram Sriram, Racuri Sudarsan, Conrad Bock, Michael Grieves, Steve Waterbury, and Jim

4 By *systems engineering* we mean any methodical approach to the synthesis and deployment of an entire system that (1) defines views of that system; and (2) manages the relationship of requirements to performance measures, constraints, components, and discipline-specific system views.

U'Ren.

7 References

[Cheng] J. Cheng, K. H. Law, *Using Process Specification Language for Project Information Exchange*, International Journal of IT in Architecture, Engineering, & Construction, 2003.

[Denno] P. Denno, M. Steves, D. Libes, E. Barkmeyer, *Model-Driven Integration Using Existing Models*, IEEE Software, IEEE, September/October, 2003.

[EIA-927] Electronic Industries Alliance, *Common Data Schema for Complex Systems, EIA 927 Version 0.2 (Rough Draft)*, Electronic Industries Alliance, April 2003.

[Express-X] ISO 10303-14 *The Express-X Language Reference Manual*, Final Draft International Standard, 2003.

[Fenves] S. Fenves, R. D. Sriram, R. Sudarsan and F. Wang, *A Product Information Modeling Framework For Product Lifecycle Management,* International Symposium on Product Lifecycle Management, Bangalore, India, July 16-18, 2003.

[Frege] G. Frege, *The Foundations of Arithmetic, Translation by J. L. Austin, Second Revised Edition*, Northwestern University Press, 1950.

[Gruber] T. R. Gruber, *A Translation Approach to Portable Ontology Specifications*, . Knowledge Acquisition, 5(2):21--66, 1998.

[Gruninger] M. Gruninger, C. Menzel, *The Process Specification Language (PSL) Theory and Application*, AI Magazine, Fall 2003.

[Guarino] N. Guarino, C. Welty, *Identity and Subsumption,* LADSEB-CNR Internal Report, 01/2001, August 21, 2001.

[Halpin] T. Halpin, *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Morgan Kauffman Publishers, San Francisco, 2001.

[Halpin2] T. Halpin, *Object-Role Modeling : an overview*, http://www.orm.net/pdf/ORMwhitePaper.pdf

[IDEF0] D. A. Marca, C. L. McGowan, *IDEF0 – SADT Business Processing & Enterprise Modeling*, Eclectic Solutions Corp, 1993

[IEEE1471] IEEE 1471-2000 *Recommended Practice for Architectural Description of Software Intensive Systems*, IEEE, October, 2000.

[ISO10027] ISO 10027 *Information Resource Dictionary System*, 1990.

[ISO10303-11] ISO 10303-11 *Express Language Reference Manual*, 1994.

[ISO10303-210] ISO 10303-210:2001 *Industrial automation systems and integration -- Product data representation and exchange -- Part 210: Application Protocol: Electronic assembly, interconnect and packaging design,* 2001.

[ISO9000] ISO 9001:2000 *Requirements for Quality Assurance,* 2000.

[ISO15926-1] ISO 15926-1 *Integration of lifecycle data for oil and gas production facitlities*, International Organization for Standards (ISO), also, http://www.iso15926.org/, 2003.

[ISO15926-2] *ISO 15926-2 EPISTLE Core Model,* International Organization for Standards (ISO), also http://www.iso15926.org/, 2003.

[Lim] T.S Lim, *Reducing the Tombstoning of Small Discrete Components,* Industry Insights, Nikkei Electronics Asia, Nikkei Business Publications, March 2003 Issue.

[KIF] Knowledge Interchange Format, http://www.csee.umbc.edu/kse/kif/, 2003.

[OWL] OWL Web Ontology Language Reference, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, February 10, 2004.

[Peak-2] R. Peak, *Characterizing fine-grained associativity gaps: A preliminary study of CAD-CAE model interoperability*, Proceedings of DETC'03, ASME 2003 Design Engineering Technical Conferences, Chicago, Illinois, USA, September 2-6, 2003.

[Quine] W. V. O. Quine *From a Logical Point of View : Nine Logico-philosophical Essays (Second Edition),* Harvard University Press, Cambridge University, 1961.

[Reber] J. W. Reber, *EIA-927: Data Standards for the Integrated Digital Environment*, Presentation at May 20, 2003 Workshop at NIST [Workshop-2].

[SUMO-1] Suggested Upper Merged Ontology, http://ontology.teknowledge.com/

[SUMO-2] IEEE P1600.1, *Standard Upper Merged Ontology Working Group* http://suo.ieee.org/ 2003.

[UML] Object Management Group, *Unified Modeling Language (UML) Specification : Infrastructure Version 2.0* , Object Management Group, September 15, 2003.

[Vila] G. Ligozat, Lluis Vila, *Ontology and Theory of Time*, http://citeseer.ist.psu.edu/ligozat98ontology.html, 1998.

[Workshop-1] *Aerospace Product Data Exchange 2003, an International Workshop*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, April 7-10, 2003 http://syseng.nist.gov/aerospace-workshop/.

[Workshop-2] *Product, Systems, and PLM workshop planning session*, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, May 20, 2003.