

Three Approaches for Knowledge Sharing: A Comparative Analysis

Mike Uschold, Rob Jasper, Peter Clark
michael.f.uschold@boeing.com
robert.j.jasper@boeing.com
peter.e.clark@boeing.com

Boeing Math and Computing Technology, P.O. Box 3707, Seattle, USA

Abstract

Our broad, overall goal is to enable cost-effective sharing of design knowledge between knowledge-based engineering software systems. To achieve this, we have identified and explored three different approaches for knowledge sharing, which we present in this paper:

- (i) Sharing services via point-to-point translation
- (ii) Neutral interchange formats
- (iii) Neutral authoring

In all of these approaches, the issue of translation between the different underlying ontologies plays a major role. These three approaches differ significantly along several dimensions, including their cost (both immediate and long-term), scale, usability, and maintainability. In this paper, we provide a description and critical assessment of each, based on one or more illustrations that used each approach. We analyze their successes and limitations, and offer some subjective advice about the circumstances under which each approach is appropriate.

1 Introduction

There are many Knowledge-Based Engineering (KBE) software systems at Boeing which contain important design and engineering knowledge. A few years ago, we started the “Neutral Representation” project (Barley *et al.*, 1997); its goal was to find ways to help preserve and reuse the knowledge that is embedded in this (KBE) software. Of particular concern was the long term retention of design knowledge, much of which is not captured at all. Of the knowledge that is captured, much of it is tied up in vendor-specific formats, making it inaccessible to other applications that may require it, and requiring that engineers be trained in multiple vendor languages. These factors also contribute to difficulties in maintenance. If the same content exists in multiple applications, then they must all be synchronized as the knowledge evolves. This is a maintenance burden, and if not attended to, different knowledge based engineering systems which should be using the same knowledge will behave inconsistently.

To address these challenges, we have studied various approaches by which knowledge sharing and reuse can be achieved, and considered some of the cost-benefit trade-offs of each. In this paper, we adopt a broad meaning of the term: ‘sharing’. We emphasize that knowledge assets at Boeing must be fully exploited. This means being used by multiple persons, across multiple applications, and in multiple contexts. We purposely ignore differences between ‘sharing’, ‘reuse’ and ‘exchange’ each of which sometimes has a specific technical meaning.¹

In this paper, we consider three approaches for knowledge sharing. For each, we describe the approach, illustrate it with one or more examples taken both from within Boeing, and from the outside, both in research and industrial

¹Distinctions include whether one or more applications use knowledge at different times (reuse) versus multiple applications using the same knowledge at the same time (sharing). A further distinction is sometimes made between a single repository, being referenced by multiple applications (pass by reference, sharing) versus information being copied and possibly translated from one application to another (exchange, pass by value). In this paper, we use the term ‘sharing’ to include all of the above approaches for exploiting knowledge assets.

contexts. We indicate what the intended benefits are, and some of the cost tradeoffs. The three approaches are:

Use of sharing services via point-to-point translation: In which two or more systems share knowledge via run-time interactions. This approach is based on a “community of experts” metaphor, in which case one system will call on another to solve a problem, rather than request the knowledge to solve it itself.

Neutral interchange formats: In which knowledge, and more generally, information, is exchanged between systems via an intermediate, “neutral” format. The exchanged information may be both items of static data, or “rules” of some kind whose primary interpretation has dynamic or behavioural properties.

Neutral authoring: In which a neutral intermediate language is used for authoring, rather than exchanging, design knowledge.

By ‘neutral’, we specifically mean with respect to target implementations. This might mean neutral with respect to specific applications, or languages used by applications. In all of these approaches, the issue of translation between the various underlying ontologies and representations plays a major role. These three approaches differ significantly along several dimensions, including their cost (both immediate and long-term), scale, usability, and maintainability. In this paper, we provide a description and critical assessment of each. We analyze their successes and limitations, and offer some subjective advice about the circumstances under which approach is or is not appropriate.

2 Three Approaches to Knowledge Sharing

2.1 Current Practice

At present, the most common approach to information-sharing is through the use of point-to-point translators, converting between different formats based on different ontologies. By “point-to-point”, we mean that there is a *direct* translation from a source format (point a) to a target format (point b). An alternate approach, that is becoming increasingly common, is to achieve translation from source to target by going through a neutral interchange format. This is sometimes referred to as a “hub-and-spoke” model. This approach consists of translating each format first into the neutral format, and then from there, out to the target format.

This latter approach is used by the STEP family of standards (Steptools Inc., 1998). STEP focuses on exchange and sharing of static models (e.g., the geometric description of a particular physical part). While STEP provides mechanisms for describing rules and constraints, their use is limited to validation of the static models being exchanged and shared. Future STEP standards plan to address general exchange of design rules and constraints.

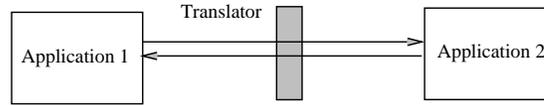
2.2 Three Approaches

In this paper, we will consider three models of knowledge sharing, described below, and illustrated in Figure 1.

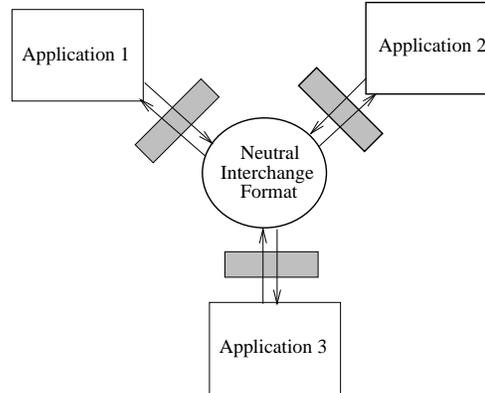
Shared Services via Point-to-Point translation: This approach entails one system making its services available to another system. Translation is required because the concepts and terms used by each system are typically not the same. If application 1 requires a service provided by application 2, the request for information, expressed in the terms of application 1 must be converted to terms that application 2 understands. Further, the response from application 2 must be converted back into terms that application 1 understands.

Note that, for this paper, we are grouping shared services with point-to-point translation as a single approach. The reason for this is historical - we performed an experiment which used this approach. However, there are

1. Shared Services via Point-to-Point Translation



2. Use of a Neutral Interchange Format



3. Neutral Authoring

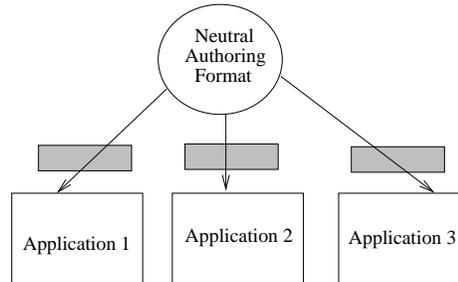


FIGURE 1: Three Different Models of Knowledge Sharing. The shaded boxes denote translators between the various representation languages.

really two essentially orthogonal issues. One is how the information is shared: the point-to-point approach, in contrast with using a neutral interchange format. The other issue is the nature of the information/knowledge being shared. Here we are talking about sharing services, as opposed to application data or knowledge in general. Any combination is possible, but we do not consider all of them explicitly.

Use of a Neutral Interchange Format: This is the ‘hub and spokes’ model for sharing information. If application J requires information that application K has, then the information must be translated from format K to the neutral format and then from the neutral format into format J. Note that this approach can be used for sharing services or for sharing application data. In this paper, we will concentrate mainly on the latter. If N is greater than or equal to four, then this approach requires fewer translators to be built. It also has the advantage that the applications can be maintained more independently. This is another in principle benefit of this approach. How and whether these benefits can be achieved will be addressed throughout this paper.

Neutral Authoring: This approach is very similar to the previous one, in that a neutral format is translated into various target application formats. However, the role of the neutral format differs – it is used for authoring rather than interchange. This means that a single authoring language is used, rather than authoring in multiple target languages. It also means that only one-way translation is required, i.e., from the neutral format, but not into it.

3 Shared Services via Point-to-Point Translation

In an ideal world, if one system requires knowledge which another system contains, then it would be easy to call upon that knowledge and make use of it as needed in a seamless fashion. One obvious approach would be for the target system to request and import the knowledge from the other system, and to process and make use of it locally (i.e., in the target system). However, there are substantial barriers to this approach. First, if the different systems use different underlying representation languages, the knowledge needs to be translated so that its dynamic behavioural properties are preserved. Second, even if the languages are the same, the knowledge in the different systems may be based on different domain ontologies, requiring that the domain terms also be translated during the knowledge transfer.

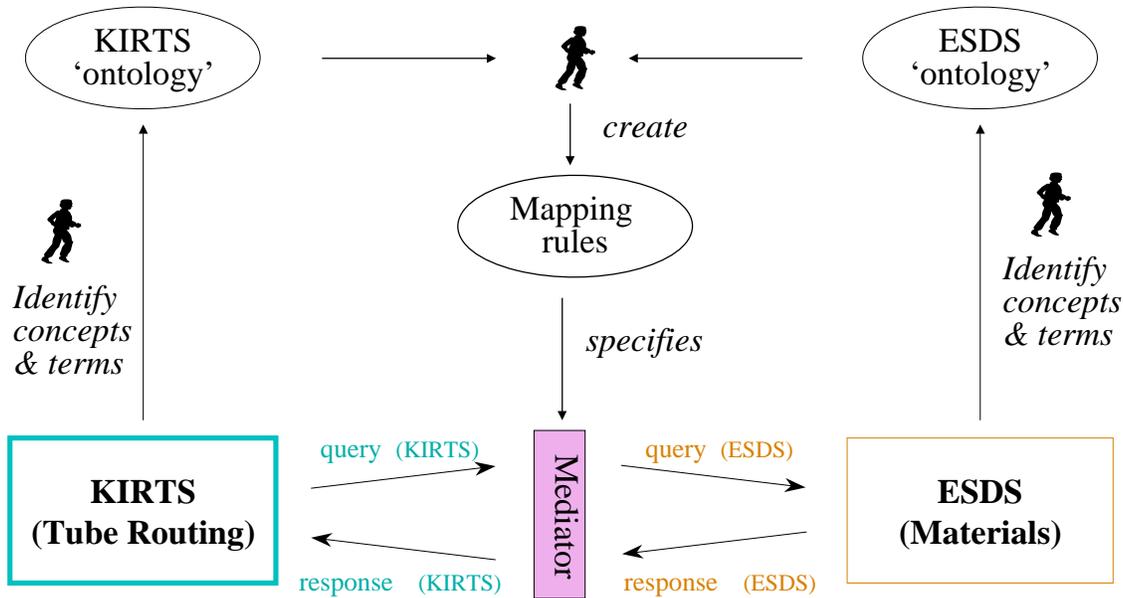
In sections 4 and 5, we will discuss two models for tackling these problems, Section 4 discussing the feasibility of this kind of knowledge-sharing by using a neutral interchange format, and section 5 by requiring the knowledge be authored in a single neutral language. However, an even simpler approach, which we discuss here, is to avoid sharing the behavioural knowledge entirely, and instead use a model of *delegation* via shared services for problem-solving. In this approach, a system requiring an answer to a particular design question will not request and import the knowledge to answer that question, but instead delegate the problem to another system capable of answering it. This simplifies the translation problem significantly. Only the questions and answers need to be translated between the different systems, not the whole knowledge base. This model is based on the metaphor of a community of experts, collaborating to solve a problem together without trying to teach each other their respective expertise.

3.1 Illustration

As part of this research, we explored this approach to link together two KBE systems in Boeing, one an expert system for (among other things) material selection, called ESDS (Dahl, 1993), and the other for generative design, based on the Genesis system (Heisserman & Mattikalli, 1998). In this context, the generative design system sometimes needs to know which material a particular tube should be made of, in order to make routing decisions. To do this, it engages in a run-time dialogue with the ESDS system, in which ESDS sends questions (eg. “which zone is the tube in?”, “what pressure will it be?”), the design system answers, and eventually ESDS sends a material recommendation.

A barrier to any attempt to share knowledge are differences between ontologies underlying different systems. The difficulty in translation is proportional to the degree of differentness. In this particular case, the two systems’ ontologies are very similar, but there are important differences too. For example, although both have the same concept ‘system category’, and both use the same term for this, the conceptual breakdown of the system into different categories differs. In addition, just because both systems use the same term, this does not mean that they refer to the same underlying concept. To find out whether they do or not requires a careful analysis of the two applications, to see what the fundamental concepts are and what terms are used to refer to them.

The main task required to link these two systems is to translate the questions and responses from the different ontologies underlying each. To achieve this, we did three things. First, we carefully analyzed each system and identified the underlying concepts and terms for each (i.e., its underlying ontology). Depending on how the system was engineered and developed, this ontology may or may not be explicit, before the analysis step. If it is explicit, it is likely to be in the form of documentation, possibly in an early system requirements document. If it is, this makes this step much easier. Second, we created a set of mapping rules that indicate how a term in one system can be mapped to a term in the other one. This set of rules specifies the requirements for a mediator, which is an explicit piece of software, separate from both systems. Creating this mediator is the third step (see figure 2).



This scenario indicates how KIRTS can access services provided by ESDS at runtime. Mapping of terminology is central to this activity.

FIGURE 2: Shared Services and Point-to-Point Translation

3.2 Scope, Costs, and Benefits of the Approach

There are several important lessons which can be drawn from our experience in this piece of work. Most significantly, this approach has a high degree of practicality: It can be made to work, and at least in the short-term is a relatively low cost, practical approach. We successfully reused knowledge that was already in another system (ESDS), without having to re-implement it from scratch, and the software was successfully built into a production system.

However, there are also significant caveats to this style of knowledge sharing. First, there is a significant challenge for maintaining such a knowledge sharing link, as the systems must be maintained in lock-step. For example, if the ESDS knowledge base changes, e.g., if terms become used in a different way, or if new terms are introduced, then the mediator must be updated. Worse yet, inconsistencies may go unnoticed until they have impacted critical applications or data. Second, direct run-time sharing increases an application's dependence on network services, which could impact performance and reliability.

Finally, there was a significant effort required to build the translator, and in some cases there was insufficient information to unambiguously determine a term from one system mapped to that of another (eg. how do air-pressure categories "low" and "high" from one system map into categories "low", "medium", and "high" in the other?). In this situation, modifications had to be made to one of the systems to request the missing information from the user. This is only possible if one or both systems were internally developed. Systems from outside vendors, or from other parts in the organization cannot be changed. This is a potentially serious barrier to sharing knowledge in this way.

The Punch Line

Question: Is Sharing Services via Point-to-Point Translation an effective way to share knowledge assets?

Answer: This approach is feasible, but there are substantial limitations.

4 Neutral Interchange Format

In the previous section, we considered knowledge sharing via point-to-point translation between two systems. However, if this approach were extended to involve many systems, it may be more appropriate to translate via a “neutral”, intermediate representation language. In the strongest case of this, we would like to transfer not just application data but design rules with behavioural properties also in this fashion. This model of knowledge sharing was the basis of the DARPA knowledge-sharing effort (Neches *et al.*, 1991), using KIF as the neutral interchange language (Genesereth & Fikes, 1992). We discuss this neutral interchange model of knowledge sharing in this section.

The neutral interchange approach requires:

1. the design of a sufficiently expressive neutral interchange format
2. the construction of two-way translators between the neutral format and each target application format

Note that, in this approach, the knowledge to be shared between various systems is authored in the original systems, not in the neutral format (we will consider authoring directly in the neutral format in section 5).

Different strategies exist for designing the neutral format. One is to make it very expressive, so that nothing is lost in the translation from the target applications to the neutral format (ie. its expressiveness covers the expressiveness of all the individual target languages). At the other extreme, the expressive power of the neutral format could be the lowest common denominator, or the intersection of the expressive power of all the target applications. This makes the creation of the translators easier, but has the disadvantage of requiring that engineers author with translation in mind, only using that restricted subset of their home formats which they know is translatable (We refer to this as “translator bias” in the authoring). In practice, some intermediary position is typically chosen between these two extremes.

This model of knowledge sharing aims to allow all applications to use information from all other applications, with several potential benefits. First, there is no need for application builders to learn a new language for authoring, since the authoring takes place in the original application formats. Second, the different systems can be maintained independently: at least in theory, the only thing requiring changing should the application language be modified would be the translators to/from one’s own format to the neutral format. Finally, there are potential savings to be gained by building fewer translators. One needs to build $O(n)$ translators instead of $O(n^2)$ which would be required if point-to-point translators were built for every pair of applications. We stress that these are *potential* benefits, which may or may not be possible or practical to realize. We discuss the actuality of these benefits later. At this point, we examine three examples of using the neutral interchange format approach, and compare and contrast them.

4.1 Illustration 1: PIF/PSL for Process Models

The Process Interchange Format (PIF) (Lee *et al.*, 1998) is a neutral interchange format for applications which build and use process models, designed so that each application can access models built using the other applications. A small team working part time over the course of a few years developed PIF, funded in part by DARPA. The Process Specification Language is a similar effort (NIST, 1999), which started independently, and was funded by NIST. PIF and PSL are in the process of being merged, so we will treat them here as if this process were complete and they are a single language. In this model of knowledge sharing, a process model resident in one application will first be translated into PIF/PSL, and then translated a second time out of PIF/PSL into another target format.

This effort distinguishes itself among most by using a rigorous logical formalism for defining and representing the core concepts required to represent processes. PIF/PSL constitutes a process ontology. It uses the syntax of the Knowledge

Interchange Format (KIF) which gives the full power of first-order logic, and has a formal semantics. The vocabulary of PIF/PSL consists of terms such as ‘activity’, ‘time-point’, ‘before’ which are used to represent processes. The semantics of the terms is given as a set of axioms, which reduces the possibility of ambiguity by ruling out incorrect interpretations of the terms and relationships. This formal definition of PIF/PSL serves as a requirements specification for the translators. Currently there are translators for IDEF3 and for ILOG. The translators being built are imperfect, insofar as they have to deal with inherent differences in expressive power of the target languages, as well as mismatches in the particular concepts that are used.

This work is in the research prototype stage, and is ongoing. PIF/PSL has successfully been used to allow the IDEF3-based ProCAP process-modeling tool to successfully exchange process information with the C++-based ILOG Scheduler, and a second pilot implementation has begun that will involve the exchange of process information between the MetCAPP process planning application and the Quest simulation application via PSL. This is expected to be completed by the end of September 1999. The authors have also created mappings from the PSL semantic concepts to EXPRESS and XML².

4.2 Illustration 2: STEP

Another example of the neutral interchange format approach being used is in the STEP standards for representing product data (Steptools Inc., 1998). The basic approach is similar to PIF/PSL, however there are also important differences. Users begin by using the language, EXPRESS, to define a schema. This schema defines the ontology for the data being exchanged. Translators for each application then read and write information according to this ontology, and expressed using a pre-defined file syntax (the “Part 21” file format).

Commercial software now exists for automatically generating the API for reading and writing from/to the neutral format. This is a significant benefit, but is a relatively small part of the translation task. The real work of translation is in deciding what calls to the API are required to translate a given data item stored in the internal data structures of an application. This remains a challenging manual effort. There are examples of STEP being used at Boeing.

4.3 Illustration 3: KIF

More ambitiously, the AI community has sought to exchange not just application data between systems, but also information whose primary interpretation has dynamic, behavioural properties, ie. which can be “executed” to perform some computation. For example, exchanging knowledge bases authored in different knowledge representation languages requires that inference behaviour be preserved. The language KIF was targeted as the neutral interchange format while the language Ontolingua was used to express the ontology used in the interchange (ie. what the non-logical symbols should be). Translation remains a difficult problem in the general case (see discussion below in § 5.2).

4.4 Scope, Costs, and Benefits of the Approach

These three examples, PIF/PSL, STEP, and KIF, all use the neutral interchange format approach, yet differ significantly in their maturity and degree of success. It is interesting to discuss the reasons for this contrast, and the dimensions along which the three different illustrations differ.

First, some of the apparent success in the STEP community may be due to sheer differences in the amount of effort applied. Each vendor supporting STEP formats devotes a significant amount of effort to obtain compliance. Further-

²These will shortly be available at <http://www.steptools.com/projects/psl/> and <http://www.nist.gov/psl/xml>. For further information, visit the PSL web site (NIST, 1999)

more, the effort is spread over each of the vendors, amounting to many hundreds of person-years of effort, one or two orders of magnitude more than devoted to the PIF/PSL and KIF efforts.

Second, and perhaps more significantly, is the nature of the knowledge being shared. STEP is currently focussed on sharing application data, corresponding to ground assertions in a logical formalism. For example, STEP can be used to exchange the geometry of a particular *instance* of a pressure tube made of titanium, but not to exchange a general rule such as: “all pressure tubes are made of titanium”. KIF, in contrast, is designed to be a highly expressive interchange format capable of expressing full first-order logic expressions, with PIF/PSL being somewhat between these two extremes. There is a general trade-off here, that more expressive the data being interchanged is, the more difficult it will be to create translators; this turned out to be a major challenge for the KIF projects. In addition, in the KIF experiments, different target languages typically only support a subset of first-order logic, and that subset is different for each target. As a result, knowledge authors need to be careful to either use only the common subset between all target languages for there to be any hope of feasible translation.

The potential benefit of requiring fewer translators may be more than offset by the expense of building the neutral format, especially where N is not very large. Experience shows that this is very time-consuming, and therefore costly. If this cost is born by public funding, or is shared among major industrial or academic consortia, then there is more hope for the cost being amortized in time among many users.

There is another potential benefit of the neutral interchange format approach: simplifying the maintenance problem, when new formats come on line, or if existing ones change. In principle, one need only be concerned with translators to and from the interchange format and one’s own application. If a new application comes on line, then once the translators are in place for the new application, then, ideally, no more work needs to be done by any of the other application developers/maintainers. This may be true as long as the interchange format remains stable, but substantial changes in target formats, may require the interchange format itself to be updated, which undermines this benefit.

Another tradeoff of this approach is that there may be more lost in two translations than in a single translation performed by a purpose-built point-to-point translator.

In light of these considerations, in terms of getting the job done on time and within budget, point-to-point translators may sometimes be preferred.

The Punch Line

Question: Are Neutral Interchange Formats an effective way to share knowledge assets?

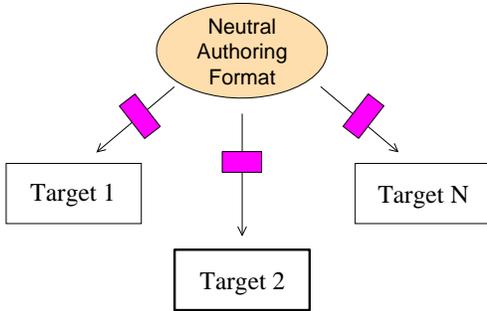
Answer: Yes, for application data, although there are limitations.

Not yet for sharing information whose primary interpretation has dynamic behavioural properties. In general, building translators in this case, it beyond the state of the art. However, this is a subject of active research.

5 Neutral Authoring

We now turn our attention to the use of Neutral Authoring as a way to share knowledge assets (see figure 3). This approach is similar to neutral interchange format approach, however, in this approach, knowledge is authored in the neutral format, and translation is only required one way, from the neutral format into target formats. Nothing is translated into the neutral format.

In both cases, there is the time-consuming task of designing the neutral format. However, because it is to be used differently, there may be different design considerations. While it might possible in principle to use PIF/PSL for authoring process models, it was not intended for that. One issue is readability, in that a format that is used only for



This approach entails designing a sufficiently expressive neutral format and building one-way translators from this format. The neutral format is used for authoring, not for interchange.

FIGURE 3: Neutral Authoring Modeling

interchange need not also be human readable. Also, because only one way translation is needed, there may be stronger arguments for designing the language to be the lowest common denominator in expressive power. We will discuss two different cases where neutral authoring was used during research here at Boeing.

There are several potential benefits of the neutral authoring approach. First, by authoring in a single format and translating into multiple target languages, we reduce dependence on particular vendor formats. In addition, we need only maintain one version of the knowledge. Together, these contribute to a potentially cheaper and more effective retention of important knowledge assets.

5.1 Illustration: The Specware Experiments

To explore this approach in detail, we performed a series of experiments using a tool called Specware (Jullig *et al.*, 1995). Specware is a tool for the specification and formal development of software, in which software is first specified using a high-level specification language (called SLANG), and then these specifications are interactively translated to executable code (in either Lisp or C++) via a formal process of refinement. In our context, SLANG provides the syntax for a neutral authoring language, and its specification refinement capabilities provides interactive support for translation to the two target languages which it supports.

In the first experiment, we started with a small, existing piece of engineering software encoded in ICAD, a knowledge-based engineering language. This was reverse-engineered, and re-expressed in a “neutral” representation, using Slang, the language of Specware (Williamson *et al.*, 1997). This included factoring the knowledge in the original software into modular SLANG “components”, representing theories for engineering concepts such as materials, simple physics, real numbers, and geometry. The SLANG components were then combined to produce a neutral statement of the problem for which software is required to solve. Finally, Specware was used to refine that specification into executable Lisp code, and the code incorporated into a larger ICAD application performing design of a simple part.

In the second experiment, we explored the feasibility of importing an existing theory from outside Boeing, rather than creating all the theories from scratch, to assess how cost-effective such reuse might be. In this experiment, the theory

we imported was the Engineering Math ontology in the Ontolingua knowledge repository (Gruber, 1993), containing knowledge about units of measurement and units conversions. The original theory was expressed in the language Ontolingua, rather than SLANG (ie. in a different specification language), and so a (manual) translation of knowledge at the specification level had to first be made. This in itself turned out to be challenging, although feasible, and our overall conclusion was that there was a net benefit in this process compared with starting from scratch ((Uschold *et al.*, 1998)). Finally, the Engineering Math theory could be combined with the other existing theories in SLANG to produce an enhanced specification of the target knowledge.

Given the technical complexity of authoring in a specification language, based on logic and category theory, and controlling the refinement process, it seemed unlikely that a wide-scale use of this approach for engineering design would be feasible in the near future. As a result, we conducted a third experiment, exploring the possibility of creating a hybrid engineering design environment. In this environment, a library of basic geometric components would be authored in a neutral language (SLANG), and then converted to software in different standard KBE languages, capable of executing and creating the geometric data which the components described. Then, design of specific parts would continue in the native KBE languages, but using this shared library, derived from the neutral representation.

The results of this third experiment were as follows. We demonstrated that this approach was feasible for some very simple parts (which included simple attributes and subparts), and developed a neutral language for describing them. We were able to translate them into two KBE languages (ICAD and AML³), and successfully loaded and executed them in the target software systems. However, this initial success was limited to using features that were essentially the same in both languages. The translation was relatively straightforward, and the difference between the AML and ICAD versions of the parts was limited to minor syntactic differences in the languages. When we attempted to include additional features (eg. reference chains and positioning), important differences arose in the way they were handled by the two target languages. To make translation of such features feasible, general theories describing them would have been required. It wasn't clear how these general theories would apply in other environments and languages.

5.2 Illustration: Ontolingua

Ontolingua, was designed as a neutral format for authoring ontologies. There are a suite of translators which convert the ontologies into the desired language (e.g. Prolog, Loom, Clips and many others). From the early literature on using ontologies and translation for knowledge sharing and reuse (eg. (Cutkosky *et al.*, 1993; McGuire *et al.*, 1993)), it was possible to get the idea that robust translators existed that could accurately translate knowledge bases to and from Ontolingua. However, there are few if any published reports describing major applications or research experiments where using these translators provided substantial benefits. In fact, these translators were quite limited, mainly being a syntactic rewrite rules, and the widespread use of knowledge sharing via fully automatic translation to/from Ontolingua still remains a long-term goal to be achieved (see (Valente *et al.*, 1999; Grosso *et al.*, 1998; Uschold *et al.*, 1998) for in depth discussions of some of these issues. In the short term, these tools may be helpful in kick-starting the translation process – i.e. the translators provide a first cut, which is then taken as a starting point for producing a translation.

5.3 Scope, Costs, and Benefits of the Approach

As with the other approaches, translation is a key to this approach, and similar difficulties arose. Translation is hard to do in general, but within certain limited domains and circumstances, translation can be accomplished. Differences between the expressive capabilities of different target languages, means that avoiding “translator bias” remains a significant challenge. This was discussed in the context of the neutral interchange format approach; it also arises when doing point to point translation. This issue manifests itself slightly differently, in this approach, since we are now authoring in a single language. In particular, it impacts on the design of the neutral format. To eliminate translator bias,

³AML: Adaptive Modeling Language

one can adopt a neutral authoring format that is the least common denominator of expressive power of the intended target languages. If the format is more expressive, and then authors will have to be careful about which features they use, knowing that they will not all be translated equally well into the different target languages. This will require using a subset of the authoring language, to ensure that the required translation is carried out effectively. Note that, this issue of translator bias seems to be unavoidable. Furthermore, it undermines the whole point of having and using neutral formats, both for authoring and for interchange.

The Punch Line

Question: Is Neutral Authoring an effective way to share knowledge assets?

Answer: It depends.

Not yet in general, due to translation difficulties. In addition, the cost of designing the neutral language is a significant obstacle. There are no widely accepted general principles for developing neutral languages, e.g. where the best place is along the expressive power continuum, given as set of target languages.

Yes, in limited circumstances, the neutral authoring approach can be cost-effective: First, if the target language can be tightly restricted, then this approach can be feasible. In our case, Specware outputs code into highly restricted (applicative) subsets of C++ and Lisp. If these are adequate for expressing the design knowledge, then this approach may be suitable. Second, if trace-ability and/or verifiability are very important, then the guarantees offered by automatic refinement tools such as Specware offers important advantages, and may make this approach cost effective. Finally, trained authors must be available to use the neutral authoring format, or the cost of training them must be limited.

The Ontolingua translators do not appear to be in widespread use. We believe this is because the general problem of translating arbitrary knowledge in an ontology is beyond the state of the art. However, In the short term, these translators may be helpful in kick-starting the translation process.

However, the existence of Ontolingua and the ontology library is still of great value. It is far easier to start with an existing ontology than to build one from scratch, even if there are no automatic translators.

6 Summary and Conclusions

Our goal in this paper is to explore different ways of achieving knowledge sharing in a cost-effective manner, and assess their benefits and tradeoffs. We identified and described three different approaches for knowledge-sharing:

- (i) Sharing services via point-to-point translation
- (ii) Neutral interchange formats
- (iii) Neutral authoring of design knowledge

Shared services seems an immediately feasible approach to knowledge sharing, but with some substantial and fundamental limitations as described earlier. Neutral interchange formats can be feasible, but it depends on the nature of information being exchanged. If the exchanged information is application data, without behavioural properties (ie. is not to be “executed” in a target environment), then neutral interchange formats can be a cost-effective approach as the current results with STEP suggest. However, this approach is currently not yet mature enough for exchanging data with behavioural properties, and building fully automatic translators in the general case is beyond the state of the art. Finally, neutral authoring similarly suffers from the difficulties of translation and the cost of language design. In general it is not yet a viable option, but there are special cases where it can be feasible, e.g. when the target languages are suitably constrained.

An Alternative to Translation

In all of these approaches, the issue of translation between the various underlying ontologies in the different languages is a major issue, and the greater the gap between the target languages, the more difficult the task. This is true whether the translations are point-to-point, or via a neutral format. However, there is an alternative approach, not reliant on translation. It is to agree on a common language standard. The underlying ontology of this language would be some blend of the ontologies of the existing languages. This ontology, and the process of creating it would likely be similar to the effort and results of creating a neutral interchange format.

A good example of this approach is the evolution of a the language (VHDL (VHDL International, 1999)) for specifying electronic circuit design. There were several competing languages for which translators were required to achieve sharing and/or inter-operation. A common language alleviates the need for translation. However, there is a cost, namely, that certain language features are no longer supported. A related tradeoff is that this may stifle creativity.

Let us see how this may apply in the case of multiple KBE languages for creating part designs. If designs need to be shared, then translators must be developed between the different representations. Translators can be very expensive to produce, and often they leave much to be desired - i.e. not everything can be faithfully translated. An alternative which may be cheaper and more effective in the long run might be to avoid translation altogether, and to have all the major players agree on a standard KBE language (e.g. Boeing, Ford Motors, etc) analogous to VHDL. Today, KBE vendors compete on various features in the language and tools, to distinguish their products. In the new marketplace, they would instead compete on the quality of the compilers and the environments. This is somewhat analogous to the experience with the introduction of Common Lisp.

Note that this is actually the neutral authoring approach in disguise. There is a single language, and it is 'translated' (i.e., compiled) into the native data structures of competing vendor systems. This shifts the burden of translation to the vendors, but it does not disappear completely. But, because such compilation is better understood than translation between high level languages, some of the problems are minimized.

There is an inherent tradeoff between the need for different expressive capabilities for different purposes, and the need to share information between systems. If sharing starts to become more important, then some sacrifices need to be made in expressive power. If we buy into any approach based on translation, then this tradeoff is manifest by using a restricted subset of a language, so that it will translate well into intended target formats. This is functionally equivalent to a loss of language features, which is exactly what happens if a standard language is adopted. There is no free lunch.

References

- Barley, M., Clark, P., Williamson, K., & Woods, S. (1997). The neutral representation project. In *Proc AAAI-97 Spring Symposium on Ontological Engineering*. AAAI Press.
- Cutkosky, M., Engelmores, R., Fikes, R., Genesereth, M., Gruber, T., Mark, W., Tenenbaum, J., & Weber, J. (1993). PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, pages 28–37.
- Dahl, M. (1993). ESDS: Materials technology knowledge bases supporting design of boeing jetliners. In *Proc 5th Innovative Applications of AI (IAAI-93)*, pages 26–33, CA. AAAI Press.
- Genesereth, M. R. & Fikes, R. E. (1992). Knowledge interchange format: Version 3.0 reference manual. Tech Report Logic-92-1, Computer Science, Stanford Univ, CA. (<http://logic.stanford.edu/kif/kif.html>).
- Grosso, W., Gennari, J., Ferguson, R., & Musen, M. (1998). When knowledge models collide (how it happens and what to do). In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management. Track: Shareable and reusable components for knowledge systems*, Banff, Alberta, Canada. See URL:

<http://ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc.html>.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220. (<http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html#ontolingua-intro>. Also see the project page at <http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/index.html>).

Heisserman, J. & Mattikalli, R. (1998). Representing relationships in hierarchical assemblies. In *Proc of Design Engineering Technical Conference (DETC'98)*. ASME.

Jullig, R., Srinivas, Y. V., Blaine, L., Gilham, L.-M., Goldberg, A., Green, C., McDonald, J., & Waldinger, R. (1995). Specware language manual. Technical report, Kestrel Institute. (<http://kestrel.edu/www/specware.html>).

Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost, G., & other members of PIF working group (1998). The process interchange format and framework. *Knowledge Engineering Review*, 13(1):91–120.

McGuire, J., Kuokka, D., Weber, J., Tenenbaum, J., Gruber, T., & Olsen, G. (1993). SHADE: Knowledge-based technology for the re-engineering problem. *Concurrent Engineering: Applications and Research (CERA)*, 1(2).

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. (1991). Enabling technology for knowledge sharing. *AI Magazine*, pages 36–56. (also see <http://www-ksl.stanford.edu/knowledge-sharing>).

NIST (1999). The process specification language project (web site). (<http://www.nist.gov/psl/>).

Steptools Inc. (1998). The ISO STEP standards. <http://www.steptools.com/library/standard/>.

Uschold, M., Healy, M., Williamson, K., Clark, P., & Woods, S. (1998). Ontology reuse and application. In Guarino, N., (Ed.), *Proc of the Int Conf on Formal Ontology in Information Systems - FOIS'98 (Frontiers in AI and Applications v46)*, pages 179–192, Amsterdam. IOS Press.

Valente, A., Russ, T., MacGregor, R., & Swartout, W. (1999). Building and (re)using an ontology of air campaign planning. *IEEE Intelligent Systems*, 14(1):27–36.

VHDL International (1999). VHDL international (web site). (<http://vhdl.org>).

Williamson, K., Healy, M., & Jasper, R. (1997). Formally specifying engineering design rationale. Technical Report ISSTECH-97-011, Applied Research and Technology, The Boeing Company.