K6 Manufacturing, Inc.
15 MacDonald Drive Nashua, NH 03062 (603)888-4669 fax (603)888-4736 Internet: kirkwood@k6mfg.com

White Paper
Design Change Vectors

Abstract

When discussing improved integration between CAD systems we generally speak in terms of translators and we measure translator quality in terms of % entities supported. Very few, if any, translators score 100%, but even if they did, there is a general perception (a correct one) that this style of integration between CAD systems would still be inadequate for many purposes.

This paper discusses a methodology for integration between CAD systems that substantially exceeds that available from translators, yet does not require CAD tools to use a common database.

Introduction

This document discusses Design Change Vectors and how they impact the design process and the CAD industry. It advocates the development and application of a product(s) for creating and operating on Design Change Vectors

Definitions

Design Change Vectors	A Design Change Vector (DCV) describes the changes between 2 versions of a CAD "file", not just what changed but also how it changed. Correctly implemented, DCV's behave much like 3-space vectors in that we can perform elementary operations on them like addition and subtraction.
CAD Database	The file format used to store CAD data. Some CAD databases don't actually use files, but it has no impact on the conclusions of this document. Based on the context of the usage, this term may mean a type/format of the database (Autocad, Unigraphics, CV, STEP) or an instance of that database type.
Downstream applications	A downstream application is one that makes use of CAD data created elsewhere. Some downstream applications add their data to the CAD data, while others store their data in a separate file. Unless it is relevant to the topic, this document assumes downstream applications store their data with the CAD data. Typical downstream application types follow: NC Programming, Finite Element Analysis, Mold flow analysis, Tolerance/Assembly analysis, Fixture Design, Cost Estimating, Bills of Material. While downstream applications use data from a CAD system, most interact with a user to create addition data associated with the CAD solid model.
Downstream organization	An organization that uses CAD data created by another organization. Typical downstream organizations follow: Manufacturing, Structural Analysis
STEP	ISO 10303 (**ST**andard for the **E**xchange of **P**roduct model data).

	A file format for exchanging product designs. It includes a wide variety of sub-formats but the most important is that for solid models.
Tight-integration	Integration between 2 applications (generally the CAD system and a downstream application) such that changes to the design via the CAD system are visible in the downstream application. Many claim that the only way to achieve tight integration is by having both applications use a common database instance. This paper describes tight integration without a common database. Other publications might also call this level of integration associativity or close-coupling.

State of the Art

Overlay comparisons

There are a variety of applications on the market today that recognize changes between versions by performing a pixelized equivalent to laying one version image on top of the other. They have some degree of acceptance with 2D outputs from CAD systems.

Delta Files

In a paper of that title by Prof. Martin Hartwick of RPI, he reported on a project to capture the changes between to CAD files. While the project was successful, it relied on every entity in the database having a persistent identifier. A persistent identifier is not generally available in the data exported from most CAD systems and so the project was not pursued further. This paper also reinforced a prevailing perception that recognizing changes required a persistent identifier on each entity. DCV disproves that notion.

K6 Manufacturing, Inc.
15 MacDonald Drive Nashua, NH 03062 (603)888-4669 fax (603)888-4736 Internet: kirkwood@k6mfg.com

Problem Statement

Design teams in most industries have a wide range of problems dealing with design changes. The following table maps out these problems.

Problem	Symptoms	Business Impact
Communicating Changes	Errors in executing changes High costs of propagating changes Major errors are explained as unusual exceptions (changes) [1].	Project costs and schedules are very difficult to manage. Minor changes seem unreasonably expensive. Even reverting to an earlier version seems to be expensive.
Processes & training neglect design changes.	see business impact	Engineering departments give very impressive demonstrations of their design tools, but project costs and schedules show little improvement [2][3]
The productivity improvement process is bogged down.	see business impact	Productivity improvement is much slower than what one might expect from the current pace of improvement in design tools. [4]

1. Changes are the rule, not the exception. Look at your company's most expensive design errors of the past 5 years. You will probably find that design changes or the inability to execute changes quickly are a critical factor in most design errors. See appendix A.
2. Much of your training comes from your vendors. That training is focused on what the tool can do, not what it can't do. It's instructive to observe that most software training departments are part of the vendors marketing organization.
3. Your processes may describe the proper handling of a design change, but were the processes and design tools developed to handle changes efficiently? Good design processes and tools must recognize that design changes are the rule, not the exception. Your design process probably requires the same steps for a change as for a new design, except that for changes, everyone has more latitude for expeditious judgment (aka. cutting corners).
4. Consider 3 modern technologies for ease of implementation.

   MRP (Material Requirements Planning) became a proven technology for reducing inventory and improving lead times in the 1960s, but there was a catch: Proper implementation required business-wide planning and implementation. Successful implementations were very slow in coming, particularly in big companies. Even today, when most companies use MRP software to manage inventory, the inventory reductions and improved lead times remain elusive for many.

   NC Machining became a proven technology for improving setup time, tooling costs, lead-time and quality in the 1960s, and it delivered. NC could be implemented in a single department.

   Computer Aided Design became a proven technology for improving throughput and reducing lead-time in the 1970s, and it delivered. CAD could be implemented in a single department.

   In the early 1980's productivity improvements from NC and CAD at the department level showed signs of leveling off. Some pundits began to call departmental advancements "Islands of automation". Everyone recognized the need for business-wide evaluation of the design/manufacturing process.

15 MacDonald Drive Nashua, NH 03062 (603)888-4669 fax (603)888-4736 Internet: kirkwood@k6mfg.com

Much of the business-wide assessment focused on selecting a unifying corporate CAD database to assure that all users of CAD data would "speak the same language".

There is general consensus that data translation is not the solution, most blame divergent geometric constructs or problems reconciling round-off errors. This is nonsense, but it's easier for a general audience to understand than the real problem. The real problem is that the most critical data, the high level relationships between the geometry and other data are left out of the translation. Many of these data relationships have no standard to define their format.

CAD/NC projects (aka CAM/CAM) began to look more like MRP projects, and positive business impact became just as rare, particularly in large enterprises. At the same time, implementation costs increased.

So what can we conclude from the preceeding chronology?

> Don't resign yourself to waiting for a business-wide solution unless there is no other choice. *Design Change Vectors offer that other choice.*
>
> Restating: Business-wide productivity improvements are great, but that's no reason to put departmental improvements on hold. If the same problem can be solved at the departmental level or the enterprise level, solve it at the departmental level.

CAD Translators and Design Changes (aka Flat-File integration)

There is a general perception that translated CAD data isn't as useful as CAD data created in the preferred CAD database. This section examines the reasons why.

The simple case (without design changes)

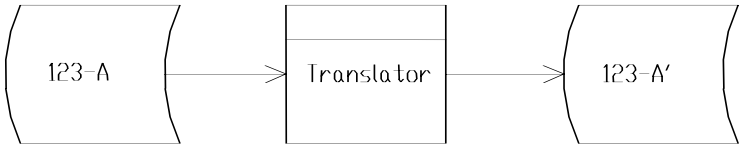


Figure 1

In this simple case, the only problems with a translator are possible problems in translating particular entities. Thus translator quality is often measured by the % of entity types that it can properly translate.

The realistic case (with design changes)

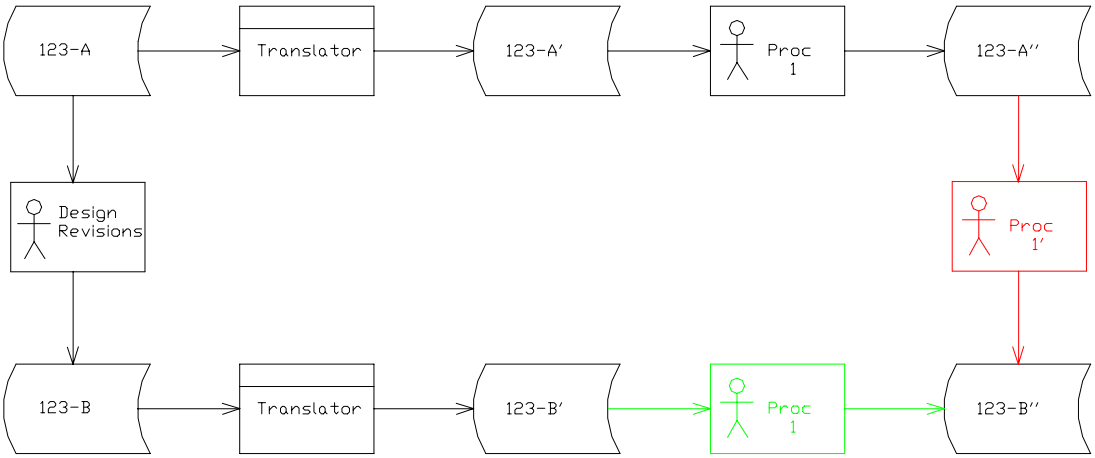


Figure 2

Let's start with the A-row of Figure 2. It's really not much more complicated than Figure 1 all we did is add a little reality. The simple case assumes we only translated part 123-A to exercise the translator. The stick-man at Proc 1 indicates that a manual process was applied to part 123-A' to create part 123-A'' (a structural analysis perhaps). The important point here is that the process added data to part 123-A' and that data makes reference to specific entities in that part.

There's also a stick-man process between part 123-A and part 123-B (Design Revisions). That symbolizes edits made by the designer.

Row B shows us the real problem arising from translators. It's the stick-man at Proc 1 shown in green. This process is repeated from scratch.

As long as we're claiming realism here, let's admit that there are common alternatives to repeating from scratch, but all those alternatives have a substantial risk of error. The most common alternative is to manually change from part 123-A'' to 123-B'' (the red stick-man marked Proc 1'). This approach, while very expedient is entirely manual and very error prone.

If a department needs a downstream application that isn't supported by the corporate database, that department's only choice today is flat-file integration (translators) shown here with all the associated risks and inefficiencies.

Common databases and design changes

When all designs and downstream applications use a common database, changes can be handled smoothly. So far business-wide CAD databases have had only spotty success in improving business profits. Some of the reasons follow:

1. Never rule-out stupidity. There are very disappointing cases where businesses have purchased the right software tools and then failed to use them properly. Many downstream organizations capable of working smartly have endured CAD data that isn't properly organized, created, or managed. Some might argue that items 2 and 3 that follow, would be classified as stupidity.
2. Common Instance vs Common Format. Very often, tight integration is only achieved when integrated applications use the same instance of the data, not just the same format.
3. Lack of discipline – Very often the productivity of one department requires some additional effort in an upstream department.
4. Vendors: Very often, downstream organizations are vendors (like machine shops) who are already committed to another database format. Even if they use the same database format, they probably are using a different instance of the design.
5. Legacy Data: It takes a long time for older data to work its way out of the system. Very often, new designs are created as variations from earlier older designs. Even when a product is radically new, many of the components are variations of older ones. Before legacy data is well purged from the business 5 to 10 years may pass and the business may be considering converting to yet another CAD database.
6. Third party software: No single CAD database has an overwhelming market share for most categories of CAD. Very often, downstream departments find the best (or only) software for certain tasks only runs with a foreign database.

How to use Design Change Vectors

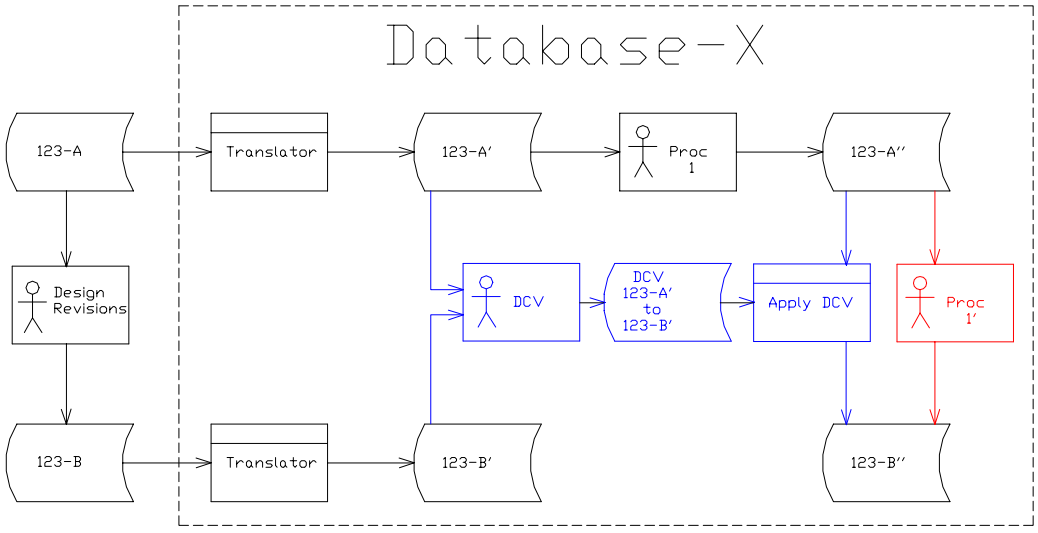


Figure 3

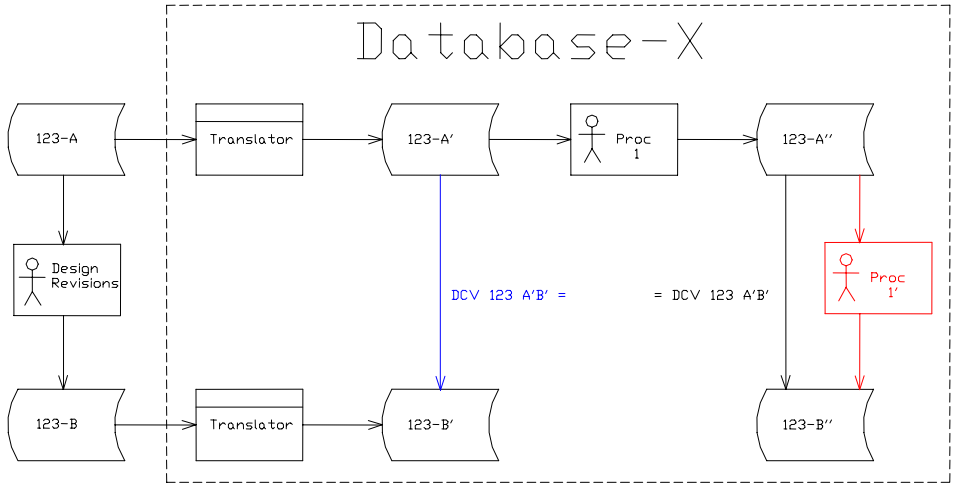
In Figure 3 we have avoided the second iteration of Proc-1 (from Figure 2). Instead we use a Design Change Vector, shown as "DCV 123-A to 123-B", in blue. The DCV process figures out the difference

between 123-A' and 123-B' saving it in the DCV file. We then apply the DCV to 123-A'' to create 123-B''.

The stick-man in the DCV process block reminds us that DCV does require some manual involvement. So did we just replace one manual process with another? No, and the reasons follow:

- DCV processing is highly automated and likely requires much less effort than repeating Proc-1 (figure 2).
- The flow chart here only shows eliminating the work for one downstream process. Generally there are multiple downstream processes. All downstream processes benefit from the one processing of the DCV.
- The DCV captures the difference between version A and B. Isn't that something that should be captured and communicated anyway?

One might observe that Figure 3 is the most complex flow diagram so far. That's only due to the notation. A notation more accomodating to DCV's follows:



Downstream Applications

A-Group

NC Programming & Structural analysis tight-integrated with one ore more CAD databases.

These applications are widely available and in most cases offer excellent tight-integration to certain CAD systems. Maintaining tight-integration with one or more CAD database formats requires a substantial engineering effort and the support of the CAD database vendors. Support from database vendors can be fickle as they may favor one downstream application vendor over another.

B-Group

Non-integrated applications: like Mold flow analysis, Tool/Fixture design, Stereo Lithography, Tolerance Analysis, Computational Fluid Dynamics

These are applications that generally don't offer a tight-integrated solution but they're valuable enough to some users that they're worth the extra effort in dealing with limited integration. Tool/Fixture design is an

additional special case, it generally runs on a CAD system, though it may not be the CAD system where the product was designed.

With DCV, the B-Group applications can more easily offer tight-integration and substantially increase their value to users.

C-Group

Struggling, customer-developed, or non-existent applications like: Process planning, Estimating, Inspection planning, Pressure vessel calculations, Solid Model Repairs, Featurizing.

These are applications that would offer good value if only they had a way to reach the entire CAD market with an integrated solution. Without an integrated solution, users often conclude that the application isn't worth the trouble. With DCV, this entire C-Group becomes viable.

Design Automation & Industrial Design

Design Automation runs upstream from the CAD system. These are systems where a user just inputs some parameters and the design is generated automatically. Perhaps there are cases where design automation does 100% of the job. More likely the results of a design automation application are imported into a CAD system for detailing.

Industrial design applications might also run upstream from the CAD system.

Industry Impact

Providers of downstream applications often have the dilemma of supporting multiple different CAD databases (AutoCAD, Unigraphics/Parasolids, PTC, CATIA, CV, etc). Their choices are as follows:

- Support one proprietary CAD database and settle for flat-file integration with all others. Since no one database represents even 40% of the market for most downstream applications[1]. This leaves the product at a serious disadvantage for most of the market.
- Support multiple proprietary CAD databases. Besides the incremental work supporting each additional database, database independence often requires serious redesign.
- Support a standard database like STEP. There's very little business justification for such a decision since there are virtually no customers using STEP as their common CAD database.

The dashed rectangle around Figure 3 reminds us that everything within the rectangle resides on one database, call it Database-X. This database finds itself with a unique business advantage:

> Downstream applications developed on Database-X can import files from other databases without the problems typically associated flat-file integration. So a developer of a downstream application has better access to greater market share if he develops for Database-X because it supports DCVs.

Conclusion: Once Database-X supports DCV capability, it will be able to attract more developers of downstream applications. Further, once downstream applications support such a database, there will be less reason to support other proprietary databases.

[1] Autocad may have a huge number of licenses but this writer is not aware of any downstream application type who's market share is largely based on Autocad.

The easy stuff

DCVs help us with many lesser tasks (that is lesser vs. cross database integration). In fact, many of the items listed here are what one might first anticipate of a tool for managing design changes.

Communicating/Verifying Design Changes

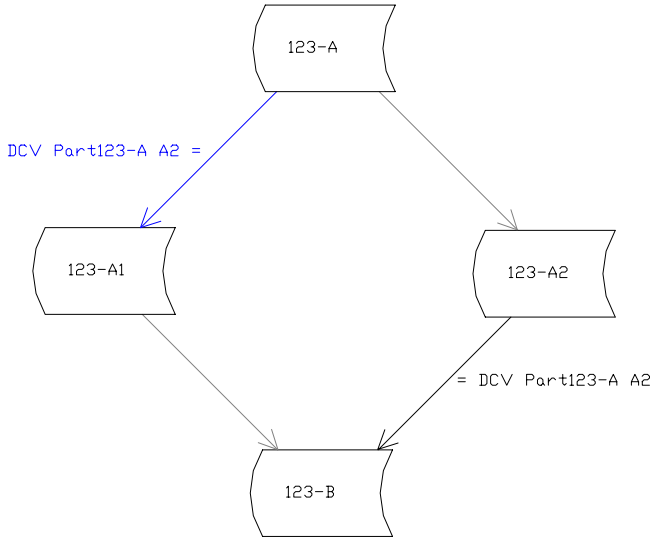
In its most elementary form, a DCV is just a description of changes applied to each geometric entity, but it is structured to reflect designer's intentions and allow added annotation information. The industry's need to better communicate design changes is amplified by the following trends:

- The ease of making changes to a CAD design (versus paper/mylar/vellum) means we get more design changes than ever before, thus we have a greater need to communicate what changed and how.
- The trend toward dimensionless CAD models makes it tougher to communicate those same changes. Before CAD, the dimensions on a drawing where the "handles" used to describe changes (ie. changed 0.874 Ø ± 0.005 to 0.8743 Ø ± 0.0002 in sector A-5). Dimensionless designs have no such handles.

Sometimes CAD systems make changes too easy. So easy, in fact, that we risk accidentally committing unintended changes. Another risk is the possibility of making tentative changes, then forgetting about them. In creating and annotating a DCV, the designer has an excellent chance of finding such errors. Further, with DCVs available to reviewers our confidence in a design/change improves further.

Teamwork

Preceding discussions of DCVs assumed that only one designer was making changes at one time. Up to now, we have assumed the work done by the other professional was adding non-design information. Well actually, it doesn't matter. Two designers can both be making changes to the same part/file at the same time, add the 2 vectors together and you have a merged design.



As with text merging (a well established technology), there is the possibility of a collision, and those cases would have to be reconciled manually. We accomplish the merge by adding DCV Part 123 A-A1 to part 123-A2. The gray vectors in the lower left and upper right only help us to visualize the merge. A DCV would only be useful there for verification purposes.

Some high-end CAD Systems/Databases might argue "we can do that". Well perhaps, but it requires careful planning up front to assure proper granularity of various parts in a design. Merging via DCV's is

much more forgiving, the only planning necessary is a brief conversation between designers just before making their changes to minimize the risk of any change collisions.

Security/efficiency

Quite often a DCV is much smaller (in bytes) than the part itself. This smaller size serves us in many ways:

- By saving the DCVs for incremental versions rather than the whole file, we save substantial disk space and/or do incremental archiving more often.
- Transmitting a DCV over a wide area network is faster and less expensive than transmitting a complete part file.
- A DCV is comparatively meaningless without one of the parts it references, so moving it over an unsecured network is less of a risk.

What about entity handles

Entity handles (also know as Object Identifiers) solve the same integration problems as DCV. DCV (under an earlier name) was originally written to transfer entity handles after flat-file integration. In fact that's what it still does. The Design Change Vector is just a different paradigm to explain what's happening.

It can't be done

If you're saying it can't be done that's actually good news. It means you do understand the scope and magnitude of this writer's claims. Suffice it to say here that it's already been done.

Conclusion

In the near term DCVs allow us to connect the proverbial "Islands of Automation", one island at a time, rather than putting everything on hold while we develop a master plan.

In the longer term DCV technology alters industry dynamics that up to now have left us with a large number of incompatible CAD data formats. DCV technology reduces the barrier between databases, perhaps enough that migration to a single database becomes possible. Further, CAD databases supporting DCV technology will have a substantial advantage in the race to become the standard CAD database for the future.

Hopefully the database we migrate to will be a standard one like STEP rather than a proprietary one.

Appendix A - Are changes that important?

Perhaps you're not convinced that design changes the major element of design costs. Consider these famous engineering failures:

Kansas City Hyatt, 1980 – An erroneous design change caused a walkway to collapse. The original design was sound.

Challenger Space Shuttle – The primary cause was bad decision to launch, but from a design perspective the fatal design flaw in the booster was known well before the launch. A design correction was in process, it just wasn't finished yet.

Three Mile Island – Another similar reactor experienced a similar problem before the problem at Three Mile Island. The necessary changes hadn't been made.

15 MacDonald Drive Nashua, NH 03062 (603)888-4669 fax (603)888-4736 Internet: kirkwood@k6mfg.com

Appendix B - Why don't the CAD databases export STEP with stable identifiers?

The STEP standards don't call for it, and it may not be an oversight. The intent is to support tight integration via Application Protocols (aka: APs) in the STEP file for each downstream application type instead. Some involved in the standards process for STEP might argue it was indeed an oversight.

The CAD database vendors who provide STEP export capability seek to be the central repository of all the CAD data. So why make extra effort, to go beyond the standard (approaching violation), to support another CAD database (STEP) seeking to replace your own CAD database as the main repository?

Appendix C - What about the STEP AP approach

The format for design data (including that data required for downstream applications) is defined in various APs (Application Protocols). The STEP specifications call for APs to be written for all the various downstream applications. The limitations of this approach are as follows:

1. STEP offers a common database format, not a common database instance. Perhaps someday a toolkit will be written/extended to allow for a shared database, but this writer is not aware of one. Thus tight-integration could only be achieved by having all designers make their updates to the same file, one at a time. Updates to copies would have to be discarded.
2. When STEP files (aka database) are the main repository for design data, the AP approach can work, but most step files are exported copies from a proprietary CAD database. As item 1 preceding explains, such copies cannot deliver tight integration. *Conceivably, a proprietary CAD database might also support importing STEP files along with any included AP data added by a downstream application, but this writer is not aware of any successful implementation like that.*
3. Downstream applications are still evolving rapidly, and many are not stable enough to lock into an AP for storing their data.
4. One downstream application would be to repair faulty translations. Are we going to have an AP for that?

Appendix D – What does a downstream application need to do?

The downstream application needs to develop tight-integration with Database-X.

There's 3 ways to do this:

1. Combine their data with the Database-X data.
2. Keep any additional data in a separate database and refer to the identifiers Database-X.
3. Copy the data with identifiers into the application's database in such a way that the value-added data from the application will not be lost. This is the way I believe downstream vendors will prefer. It let's them tight-integrate with Database-X, while keeping their own database independent. That's not just for business reasons. It's also the best way to support tight-integration to multiple CAD formats.

How does this compare to traditional tight integration?

1. It can work from a copy of the design database (a 2nd instance), so the designer and others can continue with their work.
2. Tight-integration with Database-X achieves tight-integration to <u>all</u> upstream CAD databases.