Transforms by Example - the Simpler Way to Integrate Applications

Transforms By Example (TBE) *is a completely new way of building data transform software. It greatly reduces application integration costs, while improving quality and maintainability. The method is easy to learn and easy to use. If you have been building data transforms in other ways, you can easily re-engineer them into TBE form, and switch to TBE without losing your investment. Run a trial now to see how TBE can cut your application integration costs.*

Application Integration has been a huge problem for almost forty years. The challenge of getting different applications to share information has consumed a large slice of IT budgets - with indifferent results, as is seen in the information silos of many industries.

To build an interface between two applications - or between one application and some intermediate data format - the main challenge is one of **data transformation**. You need to transform data from one format to another, while preserving its meaning.

Data transforms have been built in one of two ways:

- By writing transform code in a language such as Java or C#
- By using a mapping and transformation toolset - which tries to generate the transform automatically from mappings between source and target data formats.

Neither approach is satisfactory. Procedural code is expensive to develop, and key design decisions are buried in code - so it is hard to reuse or maintain. Mapping and transform toolkits have a reputation for limited capability - handling only the simpler cases, leaving you to fall back on code for complex requirements.

There is now a third method, better than either of these - **Transforms By Example** (**TBE**).

To build a transform with TBE, you focus on your requirement - what the transform is required to do. You build a small number of **Example Pairs**. Each example pair is an instance of the source data, together with the instance of the target data which the transform is required to produce. Typically, the same data values occur at different places in the source instance and the target instance.

If the example pairs are accurate, the Open Mapping Software **Transform By Example Toolset** does the rest. In a single step, these Eclipse-based tools:

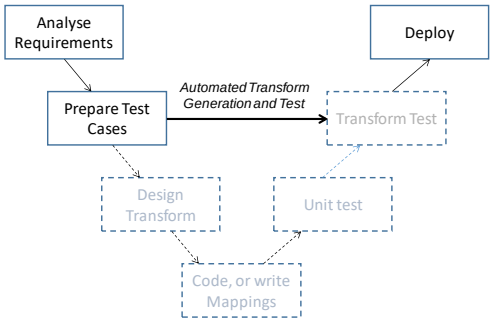
1. Check that the source and target examples have the required structure
2. Generate the transform which converts the source to the target
3. Test the transform, feeding it the source examples and other test cases to create the target outputs.
4. Compare the transform outputs with the target examples, to measure what is missing - what the transform does not yet do.
5. Output the transform in several deployable forms, such as Java code or XSLT.
6. Record a detailed trace of the generation and testing process, to help you refine the transform.

The key new element of the TBE process - generating transform software automatically from supplied example pairs - uses advanced AI-based inference techniques, which are the subject of a patent application. However, unlike many AI learning applications, large numbers of training examples are not required. An accurate transform can be generated from one good example pair, as long as that example embodies the requirement. Creating good example pairs depends only on the developer's understanding of the requirement.

Most of the work of building a transform - the design, coding, and running test cases - is done automatically by the tools. The parts you have to do - providing the example pairs - are things you would need to do in any case, to test any transform.

Because this way of developing transforms is much simpler than previous methods, and it is focused on iterative testing, it produces higher quality results.

The cost savings can be shown on a 'V' lifecycle diagram for software development:



The full 'V' diagram shows how you would develop a transform by other methods. The steps in the dashed boxes are done automatically by the TBE tools. So TBE reduces the costs of building transforms. It reduces costs as far as they can be reduced, because understanding requirements and creating test cases are activities which cannot be missed out, and TBE requires no other manual effort.

The TBE method is new, and we do not yet have extensive evidence about its cost savings. There are early indications that when source and target data representations are defined and understood, **development cost savings of 50% or more** can be expected, compared with conventional methods. For maintenance and adaptation of existing transforms, greater savings than this can be expected. Improvements in quality are at least as important as the direct cost savings.

Some points about the TBE method:

- When creating example pairs, the source part of the pair is usually extracted automatically from a source application. The manual effort is spent in creating the target part, to carry the same data as the source part. Templates are generated to help you do this.
- You will usually not create accurate and complete example pairs first time, so the tools will initially generate an incomplete transform. They have many facilities (like a good language compiler) for detecting errors and helping you correct them, to refine the transform in an iterative test-driven cycle.
- The TBE tools work in complex cases - where there are large structure differences between source and target, and where data values need to be converted between source and target.
- There will be cases (for instance, where source data formats are used inconsistently) where special transform code is required. Because the tools generate transforms as readable Java code, you can modify this code to meet special requirements.
- The TBE development process is requirement-driven and test-driven, with a rapid iterative testing cycle. The focus on requirements and testing leads to high quality transforms.
- Maintenance and reuse of transforms does not require delving into forgotten code. It involves understanding how requirements have changed, reflecting this understanding in updated example pairs. The tools do the rest.
- If you have been developing transforms in other ways, you can easily re-engineer these transforms into TBE form - by using their outputs as example pairs. You can move to TBE without losing your previous investment.

TBE can generate transforms between the following sources and targets:

- Relational databases
- Any XML or JSON whose structure is precisely defined - for instance, in XML Schema
- Healthcare data standards such as HL7 Version 2 and FHIR

Other data formats can be supported as required.

The TBE method is genuinely new, and is a real step forward. There has never before been a capability to infer transforms from examples - to generate transform software directly from its requirements. If there had been such a capability, we would all be using it now.

Transforms By Example is now the best way to control the costs of application integration, and to achieve high quality.

To learn more about Transforms By Example:

- Ask us for a free demonstration or discussion, at your premises or by webinar
- Download a demonstration pack and the TBE tools, to review demonstration projects in detail
- Read the short TBE User Guide
- Use the TBE tools to reverse-engineer some of your existing data transforms into TBE form.

For more information, see www.OpenMapSW.com or contact rpworden@me.com.

Open Mapping Software
Simply Connected