# Transforms by Example - the Simpler Way to Integrate Healthcare Applications

***Transforms By Example (TBE)*** *is a new way to integrate healthcare applications. Compared with existing methods, it reduces costs, while improving quality and maintainability. It is easy to learn and easy to use. If you have been building interfaces and 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 confirm that TBE is the best way to integrate your applications.*

The potential for integrating healthcare applications has been greatly increased by the emergence of HL7 FHIR as a common interchange standard. With or without FHIR, it is still necessary to transform healthcare data accurately between different formats. This can be costly and difficult.

Presently, data transforms are built in two ways:

- By writing transform code in a language such as Java or C#
- By using a mapping and transformation toolset - to generate the transform automatically from mappings.

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 approaches require learning a new mapping language, and have a reputation for limited capability.

There is now another method, simpler and more cost-effective than either of these - **Transforms By Example** (**TBE**).

To build a transform with TBE, you focus on your requirement. You make a small number of **Example Pairs**, which illustrate what the transform is required to do. Each example pair is an instance of the source data, together with the target instance (such as a FHIR resource or bundle) which the transform is required to produce. The same data values occur 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:

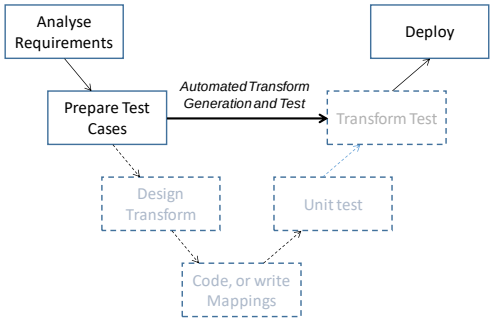
1. Check that the source and target examples have the required structure
2. Generate the transform which converts the source to the target (and the inverse transform)
3. Test the transform, giving it the source examples to create the outputs.
4. Compare the outputs with the supplied target examples, to summarise what is missing - what the transform does not yet do.
5. Output the transform in several deployable forms, including FHIR Mapping Language (a part of FHIR STU3) and Java code.
6. Display a detailed trace of the generation and testing process, to help in refining 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 need to do in any case, to understand the requirement and test the transform.

TBE is a major simplification of the transform development process. Because the process is now much simpler, the results are much more reliable.

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




Open Mapping Software
*Simply Connected*

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 instances, to carry the same data as the source instance. The tools generate target templates to help you do this.
- You will probably 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 the source and the target, and where data values need to be converted between the 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 or FHIR Mapping Language, you can modify these outputs 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 is a big 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.

TBE Transforms can be created for a wide range of source and target forms, including:

- Relational databases
- HL7 Version 2
- FHIR Bundles and Resources
- HL7 CDA
- OpenEHR
- Any XML format defined in XML Schema

FHIR profiles and extensions are supported.

The TBE tools are available licence-free to NHS organisations and to UK suppliers of Open Source healthcare software, under the NHS Code4Health initiative.

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

To learn more about Transforms By Example:

- Ask us for a free demonstration or discussion, at your premises or by webinar
- View a 30-minute webex about TBE at https://youtu.be/uyMTzib_irk
- 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 FHIR transforms into TBE form, and into FHIR Mapping Language.

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

Open Mapping Software
*Simply Connected*