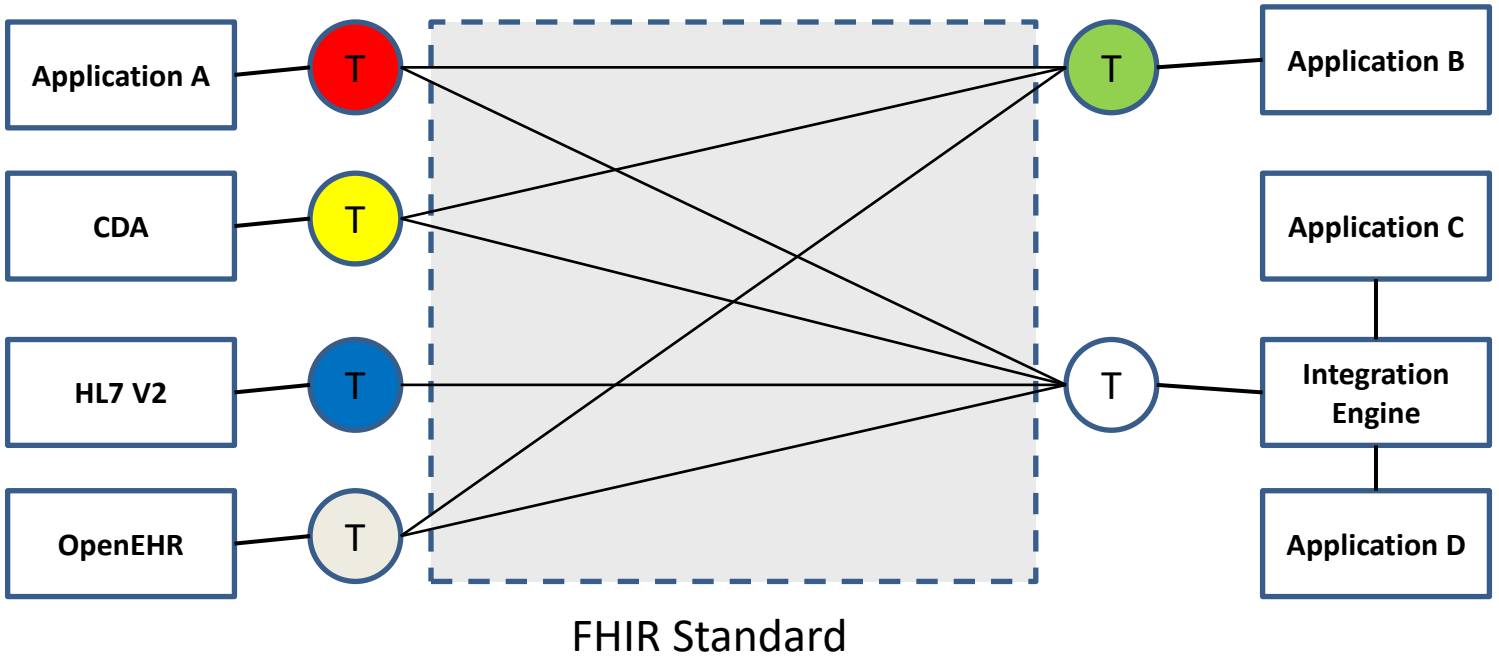
FHIR Mappings and Transforms

Grahame Grieve
Robert Worden
grahame@healthintersections.com.au
rpworden@me.com

Summary

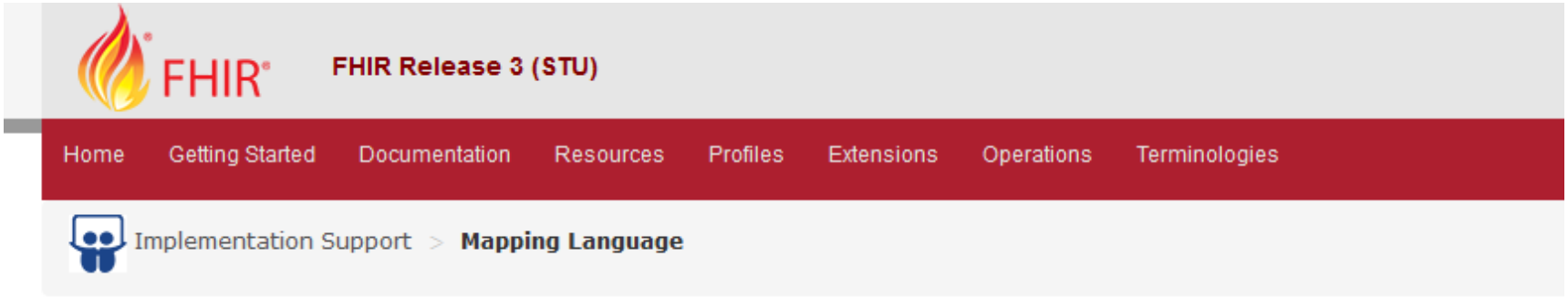
- If FHIR is to succeed as a standard for healthcare data exchange:
  - It must be a strong and usable standard in itself
  - We need many **transforms** between FHIR and other data representations
- Data transforms are expensive to build
- This is a major blocker to the uptake of FHIR
- We describe two recent advances in building FHIR data transforms:
  - FHIR Mapping Language
  - Transforms By Example
- These make it much easier to build data transforms

Interoperability with FHIR



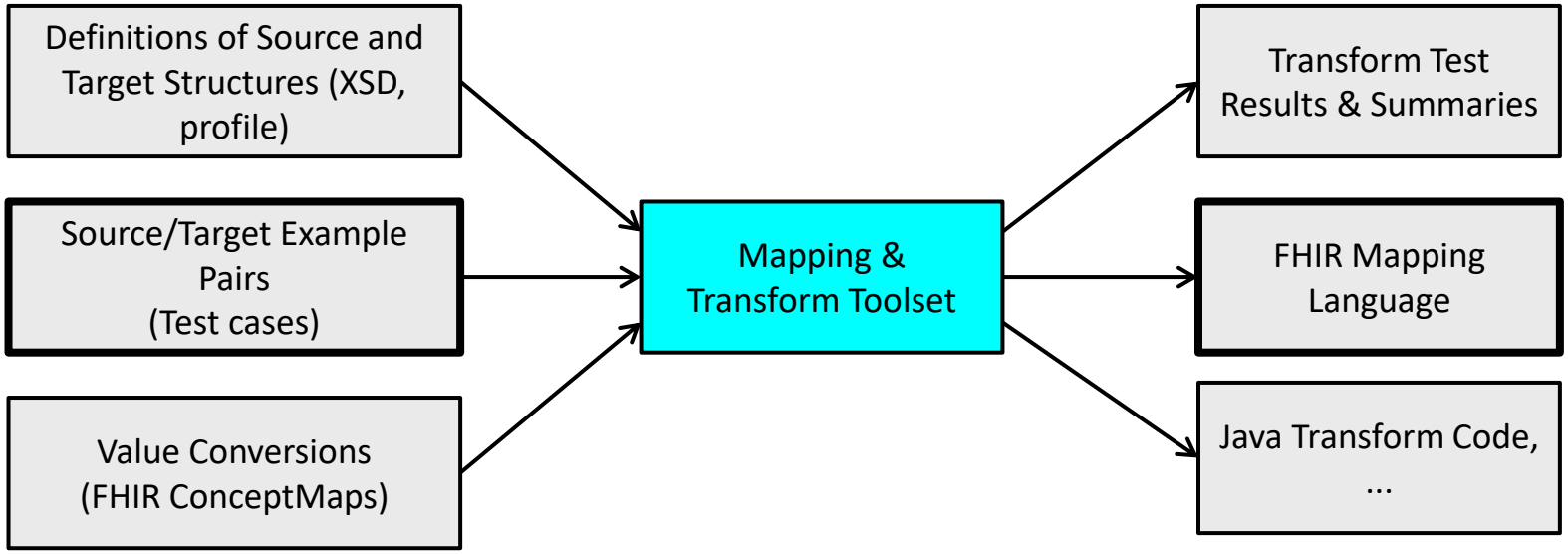
- Any application needs to work with FHIR
- Building transforms to FHIR is a big cost
- The cost holds interoperability back (e.g. INTEROpen)
- There are now better ways to build FHIR Transforms:
  - FHIR Mapping Language (FML)
  - Transforms By Example (TBE)

FHIR Mapping Language



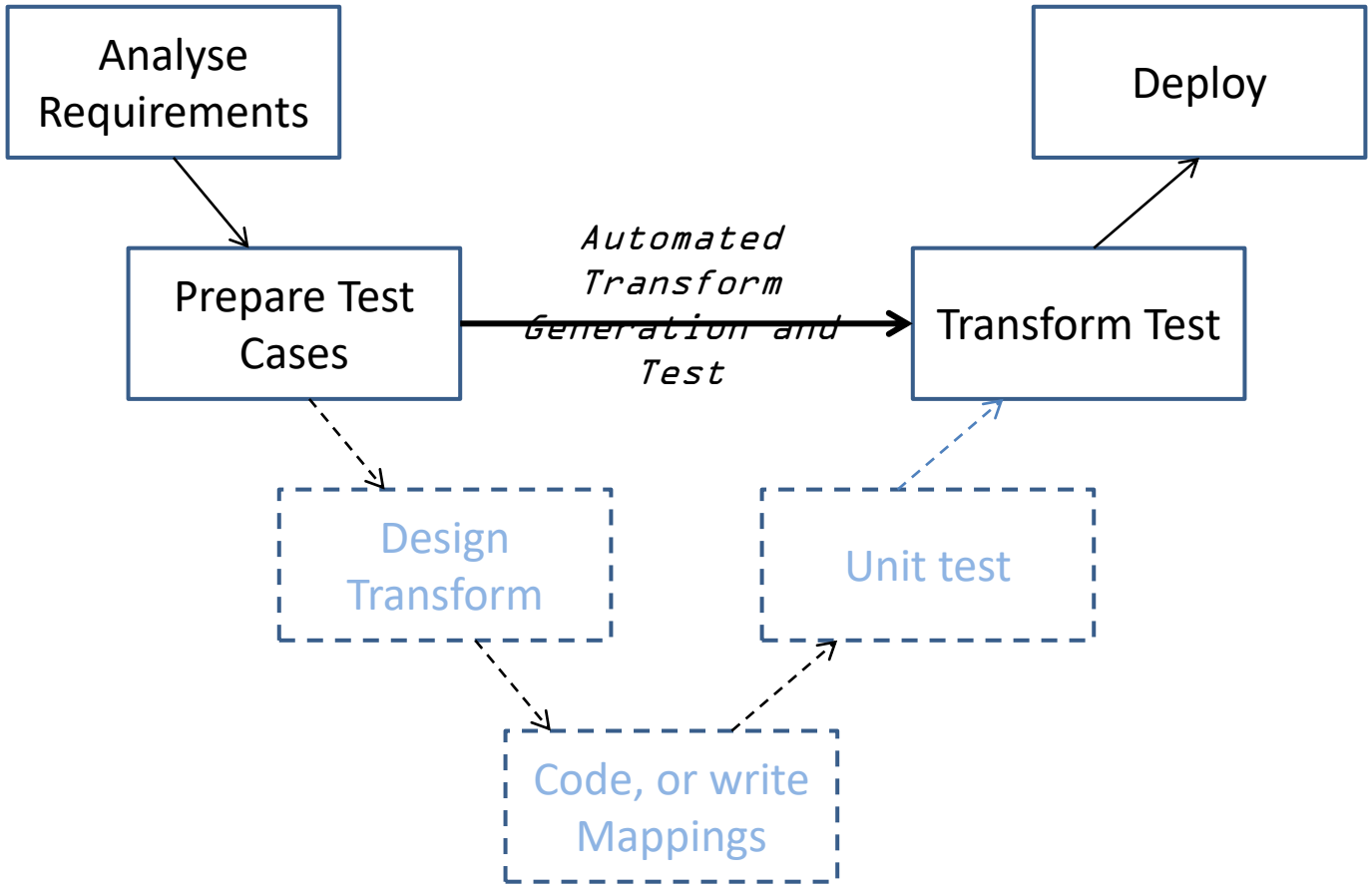
- Mappings are text files (or StructureMap + ConceptMaps)
- Declarative and compact
- A Java mapping engine exists
- Early stages of adoption
- Learning curve for the language

Transforms by Example (TBE)



- You don't need to write transform code, or learn a new mapping language
- You do need to create test cases:
  - Source/target XML pairs (1 or more)
  - Conform to source and target structures
  - Illustrate the data transforms needed
- Provide ConceptMaps for value conversions
- The tools generate and test the transform
- This is a big cost-saver

Shortening the Transform Development Lifecycle



Demonstration of FML + Transforms by Example

```
Demo case:

HL7 V2 message (ADT)
to FHIR Bundle (with Patient resource)
```

Three demonstration projects are available:

- HL7 V2 to FHIR
- Relational database to FHIR
- OpenEHR Composition to FHIR

For current status, contact rpworden@me.com.

Benefits of FML/TBE

1. Quickest way to develop transforms:
   - The only manual steps are creating the test cases, and the value conversions
   - You'll have to do that anyway

2. No new skills required:
   - No need to understand a mapping language
   - Can understand transforms in generated code
   - Easy way to learn FML

3. Generated code is flexible:
   - Can edit the generated code to do what you like
   - Use standard Java and C# tools (logging, debugging,...)

4. Test-driven development
   - Automated testing process
   - Rapid testing cycle
   - Gives high-quality, reliable transforms

5. Can reverse-engineer existing transform code (using its test cases as examples)

Enhancing Support for FML- A FHIR Foundation project

Delivering and evaluating:

- Transforms generated from examples (TBE)
- Eclipse-based Mapping toolset
- Graphical Mapping Editor
- Choice of ways to view mappings
- Transform Testing & Validation engines
- Choice of runtime transform engines
- Generated transform code (Java, C#)

For further information see the paper:

‘FHIR Mappings and Transforms’
Grahame Grieve and Robert Worden

Ways to Understand (Generated) Mappings

FHIR Mapping Language:

```
then {
 r_CodeableConc : for s_data.items as s_items1 where archetype_node_id = "at0002" and xsi_type = "ELEMENT"
 make t_resource.substance as t_substance , t_substance.coding_SNOMED_CT as t_coding_SNOME  then {
   p_filler1 : for s_items1.value as s_value  then {
     p_display : for s_value.value as s_display make t_coding_SNOME.display = s_display
     p_filler2 : for s_value.defining_code as s_defining
      then {
       p_code : for s_defining.code_string as s_code
       make t_coding_SNOME.code = s_code
     }
   }
 }
```

Java Code:

```
/**
 * @param stack - source path: composition.content.items.data
 * @param t_resource - reached by target path: Bundle.entry.resource
 */
private void r_CodeableConc(List<Element> stack, AllergyIntolerance t_resource)
{
    Element s_data = stack.get(3);
    for(Element s_items1 : namedChildElements(s_data,"items"))
    if ( pathTest(s_items1, "@archetype_node_id", "at0002") &&  pathTest(s_items1, "@xsi:type", "ELEMENT") &&  pathTest(s_items1, "nam
    {
        CodeableConceptDt t_substance = new CodeableConceptDt();
        t_resource.setSubstance(t_substance);
        CodingDt t_coding_SNOME = new CodingDt();
        t_substance.setCoding(t_coding_SNOME);

        List<Element> stack1 = push(s_items1,stack);
        p_filler1(stack1, t_coding_SNOME);
    }
}

/**
 * @param stack - source path: composition.content.items.data.items
 * @param t_coding_SNOME - reached by target path: Bundle.entry.resource.substance.coding
 */
private void p_filler1(List<Element> stack, CodingDt t_coding_SNOME)
{
    Element s_items1 = stack.get(4);
```

Next Steps

- We need to create **Case Studies** – comparing FML/TBE with hand coding of transforms
- You can help us create those case studies, using FML/TBE tools
- e.g: reverse engineer your existing transforms to FML
- The Beta Mapping Toolset is available now.
- FML/TBE can give big cost savings
- For the FML/TBE Toolset and Demo projects, contact rpworden@me.com