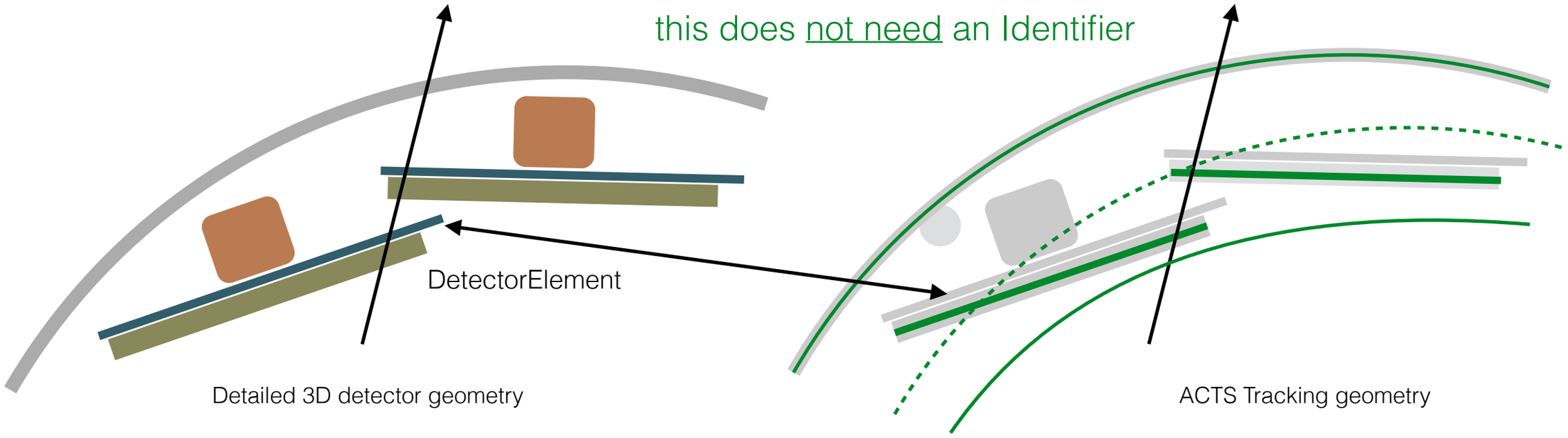
Identifier free acts

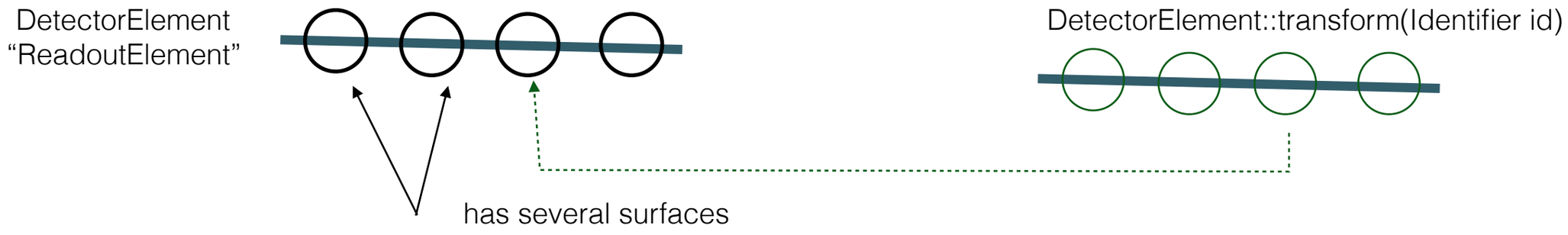
A. Salzburger

Identifier in Surfaces

existence of **Identifier** in Surface class is an ATLAS artifact



this does needs an Identifier (e.g. ATLAS TRT)



Identifier in Surfaces - fix

on Detector SW side make a distinction between
“ReadoutElement” and “DetectorElement”

Identifier in Measurements - fix

already templated:-)

```
template <typename identifier_t, ParID_t... params>
class Measurement
{
  // check type conditions
  static_assert(std::is_copy_constructible<identifier_t>::value,
                "'Identifier' must be copy-constructible");
  static_assert(std::is_move_constructible<identifier_t>::value,
                "'Identifier' must be move-constructible");
  static_assert(std::is_copy_assignable<identifier_t>::value,
                "'Identifier' must be copy-assignable");
  static_assert(std::is_move_assignable<identifier_t>::value,
                "'Identifier' must be move-assignable");
```

Identifier in DetectorElementBase - fix ?

most difficult occurance, because ideally you would like to **identify()** a DetectorElement without casting

Step 1
make DetectorElementBase Identifier free

Step 2
allow it to be exchanged by experiment SW at compile time

Step 3
alternatively allow it to be extended by experiment SW at compile time

```
#include <memory>
#include <vector>
#include "Acts/Utilities/Definitions.hpp"

#ifdef ACTS_DETECTOR_ELEMENT_BASE_EXTENSION
#include ACTS_DETECTOR_ELEMENT_BASE_EXTENSION
#endif

namespace Acts {

class Surface;

/// @class DetectorElementBase
///
/// This is the default base class for all tracking detector elements
/// with read-out relevant information. It provides the minimal interface
/// for the Acts proxy mechanism for surfaces, i.e. surfaces in the
/// Tracking geometry representing actual detection devices
///
/// The default base class can be exchanged at compile time with a
/// more extended version using the
///
/// ACTS_DETECTOR_ELEMENT_BASE_REPLACEMENT
///
/// a possible interface extension can be done with
///
/// ACTS_DETECTOR_ELEMENT_BASE_EXTENSION which has to
/// define ACTS_DETECTOR_ELEMENT_PARENT
///
#ifdef ACTS_DETECTOR_ELEMENT_BASE_EXTENSION
class DetectorElementBase : public ACTS_DETECTOR_ELEMENT_PARENT
#else
class DetectorElementBase
#endif
{
public:
  /// Constructor
  DetectorElementBase() {}
```

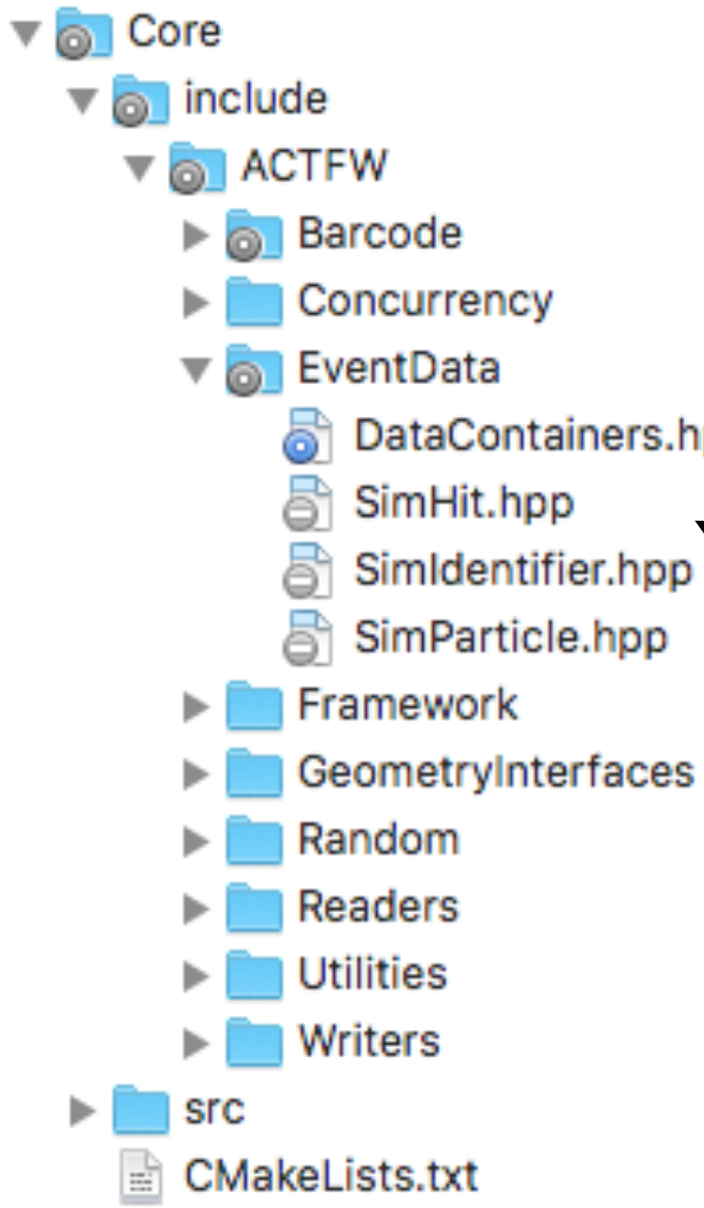
Does it work ? ACTS-486_Make_Identifier_free

Yes:
1) acts-core/Core is now Identifier free (and the **Acts/Utilitilies/Identifier.hpp** is deleted)
2) acts-core/Tests (for Measurements) just uses (as standard)
   typedef unsigned long long Identifier;

3) acts-core/Plugins uses in default
   typedef unsigned long long Identifier;

But can be overwritten and/or extended (see next slides)

Application in acts-framework



For bringing back all the acts-framework functionality following the

a) Extrapolation->Propagator change
b) Fatras new style (next meeting)

I had to sort out the **truth handling**

Step 1
make **acts-fatras** particle and hit agnostic (templated)

Step 2
Introduce SimHit/SimIdentifier/SimParticle in **acts-framework**

Application in acts-framework

Step 3

Make the SimIdentifier aware of the SimParticle

```
class SimIdentifier
{
  public:
    typedef unsigned long long  identifier_type;
    typedef long long identifier_diff;

    /// Default constructor
    SimIdentifier();

    /// Constructor from identifier_type
    ///
    /// @param value is the identifier value
    explicit SimIdentifier(identifier_type value);

    /// Constructor from identifier_type
    ///
    /// @param value is the identifier value
    SimIdentifier(identifier_type value,
                  std::vector<const SimParticle*> truthParticles );
```

Application in acts-framework

Step 4
build **acts-framework** with submodules (**acts-core**, **acts-fatras**) and declare the plugins

```
# Use the framework identifier instead of the bare ACTS one
add_definitions(-DACTS_CORE_IDENTIFIER_PLUGIN="${CMAKE_CURRENT_SOURCE_DIR}/Core/include/ACTFW/EventData/SimIdentifier.hpp")
# Use the framework detector element parent on top of the ACTS one
add_definitions(-DACTS_DETECTOR_ELEMENT_BASE_EXTENSION="${CMAKE_CURRENT_SOURCE_DIR}/Detectors/Common/DetectorElementParent.hpp")
```

ACTFW-112_Fatras_New_Propagator

(to be completed today)