New StatusCode

Frank Winklmeier
University of Oregon

Gaudi Developers Meeting
29 November 2017

if (sc) ...

Summary from Gaudi Workshop

- Replace `long` with `std::error_code`
- Discussed changing integer values to more standard-conform

```
enum { FAILURE=01, SUCCESS=10, RECOVERABLE=2 };
```

Survey of existing code revealed some surprises (to me)

- Note: Current StatusCode defines:
- This results in the following equivalences:

```
operator long() const { return getCode(); }
bool isFailure() { return !isSuccess(); }
```

```
sc.isSuccess()      ⇔   sc==StatusCode::SUCCESS
sc.isFailure()      ⇔   sc!=StatusCode::SUCCESS
if (sc)             ⇔   sc!=StatusCode::FAILURE
if (!sc)            ⇔   sc==StatusCode::FAILURE
```

In a binary SUCCESS/FAILURE world this is all fine, but....

```
StatusCode sc( StatusCode::RECOVERABLE );
sc.isFailure() : true
if (sc)        : true
```

- Our code-base is full with "`if (sc)`". What was the author's intend?
  - Was the author aware of the above?
  - Or was it intended as short for `sc.isSuccess()` ?

- Corollary
  - `StatusCode(StatusCode::RECOVERABLE).isFailure() == True`
  - `sc.isFailure()` **not** equivalent to `sc==StatusCode::FAILURE`

if (isFailure())

Anybody remember the history of this change?

- The comment suggests isFailure() used to be implemented as
  `return m_code==StatusCode::FAILURE`
- Probably best to keep the current behavior

```
56    /** Test for a status code of FAILURE.
57     * N.B. This is a specific type of failure where there aren't any more
58     * appropriate status codes. To test for any failure use :
59     * if ( !StatusCode.isSuccess() ) ...
60     */
61    bool isFailure() const { return !isSuccess(); }
```

Comparison operator `==`

In the current 1D StatusCode world the following two are equivalent:

- `sc.isSuccess()`
- `sc == StatusCode::SUCCESS`

In the new 2D StatusCode world this is no longer the case

- `sc.isSuccess()` asks the category if the given code is considered success
- `sc == StatusCode::SUCCESS` checks if code value and category are the same
  - Note: `StatusCode::SUCCESS` is a concrete StatusCode from the default category

Survey

- In Gaudi there is only one place where `sc==StatusCode::FAILURE` is used (ApplicationMgr.cpp)
- But in ATLAS we have a couple of hundred...
- `==StatusCode::SUCCESS` could be trivially replaced with `isSuccess()`
- But not `==StatusCode::FAILURE` as this would then include `RECOVERABLE`

Conclusion

- Treat 0(FAILURE) and 1(SUCCESS) as special codes and always ignore category (e.g. in `operator==`)

Status of changes

Complete set of changes in Gaudi!514

- https://gitlab.cern.ch/gaudi/Gaudi/merge_requests/514
- Together with a few points for discussion...

Experience from ATLAS

- Started implementing the necessary changes in ATLAS code base for this MR
- No show-stoppers found, migration trivial
- But explicit type conversion already uncovered a few potential bugs
  - e.g. found this in our tracking code

```
StatusCode sc = process(*lay, 0).isSuccess();
if (sc.isSuccess()) {
   …
}
else if (sc.isRecoverable())
```

```
TrkDetDescrTools/src/LayerMaterialProvider.cxx:63:61: error: conversion from 'bool' to non-scalar
                                                            type 'StatusCode' requested
                  StatusCode sc = process(*lay, 0).isSuccess();
                                  ~~~~~~~~~~~~~~~~~~~~~~~~~~^~
```