Requirement Refinement Meeting
openPASS

12.02.2020 – René Paris, Reinhard Biegel on behalf of BMW AG
Design Premise

- Try to follow best practices of open source

- Apply DRY principle on paths
  - or keep paths short - the old windows issue
  - no more openpass_this/openpass_that/algorith_m_this/algorith_m_this_and_that/...

- Reflect architectural changes
  - or simply moving stuff around
### Cleanup Directory Structure

**Proposed Structure**

```
root (.)
  • doc
  • gui
  • sim

  • license.md
  • notes.md
  • codeofconduct.md
  • readme.md

common items gui/sim
  • doc
  • changelog.md
```

- **Discussion about splitting**
- **What does Eclipse really need?**
- **Requirements GUI?**

- **Split into USER / DEV**
- **USER doc goes here**
- **Integration within hosted services (e.g. ReadTheDocs)**

- **DEV Doc goes there**

- **Focus on sim right now**
Cleanup Directory Structure

Proposed Structure

- docs
- deps
- include
- src
- tests
- contrib
  - templates
  - examples
- changelog.md
- .clang-format
- .gitignore

- External dependencies (aka ThirdParty)
  - Reference only
- The big public interfaces (e.g. WorldInterface)
- Former openpass_source_code
  - see following slides
- Former openpass_resources
- Potential helper scripts
- Not real code, but related

Does this make sense for everybody?

Shall we simply propose a format?
Cleanup Directory Structure

Proposed Structure

./sim/src

- common
- components
- core
  - common
  - master
  - slave
- CoreModules
- Interfaces
  - fakes/gmock?

- See Class Name Discussion
  - AlgorithmAutonomousEmergencyBrakingImplementation
- Former CoreShare
- Former OpenPassSlave
  - see next slides
- Integrated into slave
  - see next slides

Discussion CoreModules

How do we integrate fakes/mocks in the future?
**Cleanup Directory Structure**

**Proposed Structure**

```plaintext
./sim/src/core/slave
  • common
  • bindings
  • framework
  • importer
  • modelElements
  • modules
    • eventDetector
    • manipulator
    • spawnpoint
    • observer
    • world
    • stochastic

./modules/eventDetector
  • common
  • collisionDetector
  • conditionalEventDetector

./modules/spawnpoint
  • common
  • preRuntimeSpawnPoint
  • runtimeSpawnPoint

./modules/observer
  • common
  • observation_log
```

- **Former Interfaces**
- Dynamic Library Interfacing stuff
- Often confused with C++ interfaces

- A discussion on its own, but right (now) where it belongs

- Right now mixture of *Networks, Single Instances and Factory Classes*
**Cleanup Directory Structure**

**Proposed Structure**

- tests
  - common
  - contrib
  - fakes
    - gmock
  - unitTests
  - integrationTests
  - endToEndTests

- Class Level Tests
- Mirroring the src structure
  E.g. `/core/slave/commandLineParser_Tests.cpp`

- Everything, which needs external resources (e.g. import example.xml)

- Clamping the whole simulator without GUI
ExampleComponent v3:

algorithmComp
- algorithmComp.xml
- algorithmComp.pro
- algorithmComp.cpp
- algorithmComp.h
- algorithmCompGlobal.h

- src/ comp.cpp
- src/ comp.h

- DLL Interface
- Metainfo
- The Implementation (DRY)

Is it ok if we integrate *global in the header?

Do we really need that?

```c
#if defined(ALGORITHM_COMP1_LIBRARY)
#define ALGORITHM_COMP1_SHARED_EXPORT Q_DECL_EXPORT
#else
#define ALGORITHM_COMP1_SHARED_EXPORT Q_DECL_IMPORT
#endif
```
Continuous Integration
Repository for Build Automation Scripts

Windows

Docker Build Plan

Linux

Build Plan

BAZEL

CONAN

CMAKE

QMAKE

GitLab

Gerrit/GIT

Jenkins

Points of Discussion
Current View

- Repository for Build Automation Scripts
- BAZEL
- CONAN
- CMAKE
- QMAKE
- Docker Build Plan
- Windows
- Linux
- GitLab
- Gerrit/GIT
- Jenkins
- ProtoBuf?

Current View