13.01.16

Microservices
An Example of Usage in IGT

Bug Squashing Talk
Esther Wild and Matthias Eisenmann

dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for A Life Without Cancer

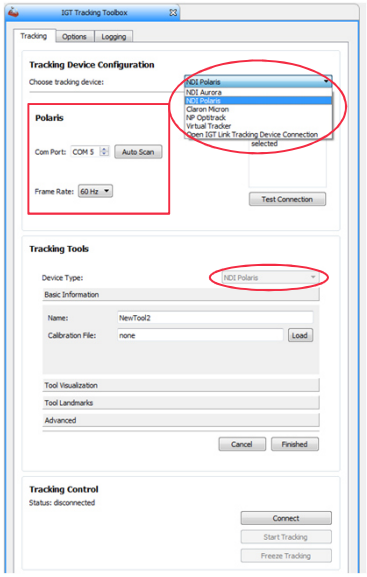
dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for A Life Without Cancer

Esther Wild
E 131
13.01.16 | Page 3

Problem in IGT

- Different tracking devices
- Implementation in external projects (e.g. MBI)
- Availability in different plugins (e.g. Tracking Toolbox)



Esther Wild
E 131
13.01.16 | Page 4

Current state

- Enum for device identification is hard coded

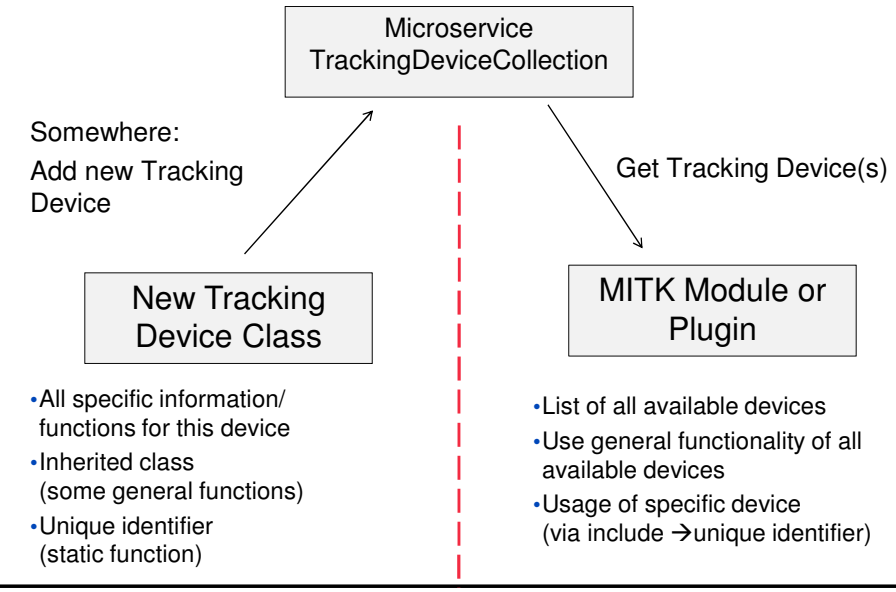
```
if (m_Controls->m_trackingDeviceChooser->currentIndex()==0) //NDI Polaris
{
  AddOutput("<br>NDI Polaris selected");
}
else if (m_Controls->m_trackingDeviceChooser->currentIndex()==1) //NDI Aurora
{
  AddOutput("<br>NDI Aurora selected");
}
else if (m_Controls->m_trackingDeviceChooser->currentIndex()==2) //ClaronTechnology MicronTracker 2
```

- All devices mixed in single document

```
  int portPolarisWin,portAuroraWin,portTypePolaris,portTypeAurora;
  propList->Get("PolarisPortWin",portPolarisWin);
  propList->Get("AuroraPortWin",portAuroraWin);
  propList->Get("PortTypePolaris", portTypePolaris);
  propList->Get("PortTypeAurora", portTypeAurora);
  propList->Get("MTCalibrationFile",m_MTCalibrationFile);
  propList->Get("SelectedDevice",SelectedDevice);
```

Esther Wild
E 131

Our solution



Esther Wild
E 131

How to write a Microservice – Header


```
#include <mitkServiceInterface.h>
#include <usServiceRegistration.h>
#include <MitkIGTExports.h>
#include "mitkTrackingTypes.h"
#include "mitkTrackingDeviceTypeInformation.h"

class MITKIGT_EXPORT TrackingDeviceTypeCollection {
public:
    TrackingDeviceTypeCollection();    ~TrackingDeviceTypeCollection();

     virtual void RegisterAsMicroservice();
     virtual void UnRegisterMicroservice();

    void RegisterTrackingDeviceType(TrackingDeviceTypeInformation* typeInformation);
    TrackingDeviceTypeInformation* GetTrackingDeviceTypeInformation(TrackingDeviceType type);
    std::vector<std::string> GetTrackingDeviceTypeNames();

private:
    std::string m_Name;
    us::ServiceRegistration<TrackingDeviceTypeCollection> m_ServiceRegistration;
    std::vector<TrackingDeviceTypeInformation*> m_TrackingDeviceTypeInformation;
  };
MITK_DECLARE_SERVICE_INTERFACE(mitk::TrackingDeviceTypeCollection,
                               "org.mitk.services.TrackingDeviceTypeCollection")
```

This macro declares our class as a Microservice

Esther Wild
E 131

How to write a Microservice – cpp part 1


```
#include "mitkTr..TypeCollection.h"  #include <usGetModuleContext.h>    #include <usServiceProperties.h>
#include "mitkUIDGenerator.h"        #include <usModule.h>              #include <usModuleContext.h>

void RegisterAsMicroservice(){
  us::ModuleContext* context = us::GetModuleContext();

  m_ServiceRegistration = context->RegisterService(this, props);
}

void UnRegisterMicroservice(){
  if (m_ServiceRegistration != NULL) m_ServiceRegistration.Unregister();
  m_ServiceRegistration = 0;
}
```

Service Registration

Esther Wild
E 131

Activator

- Difficulties:
  - Standard devices should always be available and only be registered once
- Solution:
  - Use Module activator and add devices when loading

```
#include <usModuleActivator.h>   #include <mitkTrackingDeviceTypeCollection.h>

class IGTActivator : public us::ModuleActivator
  {
    public:
      IGTActivator();
      virtual ~IGTActivator();
      void Load(us::ModuleContext*) override;
      void Unload(us::ModuleContext*) override;

    private:
      TrackingDeviceTypeCollection m_DeviceTypeCollection;

  };
```

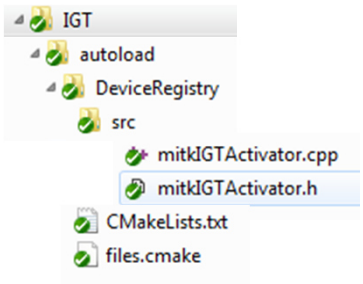
Best place to hold Microservice: Private member in Module Activator

Activator – cpp


```
#include "mitkIGTActivator.h"        #include "mitkNDIAuroraTypeInformation.h"

  void IGTActivator::Load(us::ModuleContext*)
  {
    m_DeviceTypeCollection.RegisterTrackingDeviceType(
                                          new mitk::NDIAuroraTypeInformation());

    m_DeviceTypeCollection.RegisterAsMicroservice();
  }

  void IGTActivator::Unload(us::ModuleContext*)
  {
    m_DeviceTypeCollection.UnRegisterMicroservice();
  }

US_EXPORT_MODULE_ACTIVATOR(mitk::IGTActivator)
```


Why you need to create an own module

- Problem:
  - Static variables are not initialized on Module loading
- Solution:
  - Include Activator in an own Module



CMakeList:

```
MITK_CREATE_MODULE(
IGTDeviceRegistry
  INCLUDE_DIRS
    PRIVATE src/DeviceRegistry
  DEPENDS PRIVATE MitkIGT
  WARNINGS_AS_ERRORS
  AUTOLOAD_WITH MitkIGT
)
```

MitkCore is also possible, but be careful if Qt dependent!

Esther Wild
E 131

How to use a MS in a Module


```
#include "mitkTrackingDeviceTypeCollection.h"
#include <usModuleContext.h>
#include <usGetModuleContext.h>


us::ModuleContext* context = us::GetModuleContext();

std::vector<us::ServiceReference<mitk::TrackingDeviceTypeCollection> > deviceRefs =
   context->GetServiceReferences<mitk::TrackingDeviceTypeCollection>();

// error handling (see Doxygen documentation)

const us::ServiceReference<mitk::TrackingDeviceTypeCollection>& deviceRef =
   deviceRefs.front();

mitk::TrackingDeviceTypeCollection* deviceTypeCollection =
  context->GetService<mitk::TrackingDeviceTypeCollection>(deviceRef);

// do something with the service
for (auto name : deviceTypeCollection->GetTrackingDeviceTypeNames()) {
       . . .
}

context->UngetService(deviceRef);
```

Esther Wild
E 131

How to use a MS in a Plugin – part 1

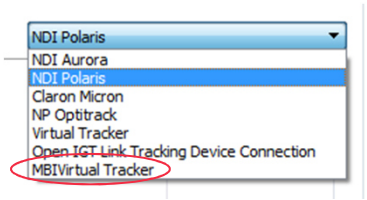
- Problem: No Module context
- Solution: Use Plugin context and save it on Plugin activation

```
#include <ctkPluginActivator.h>

class PluginActivator :   public QObject,
                          public ctkPluginActivator
{
  Q_OBJECT
#if QT_VERSION >= QT_VERSION_CHECK(5, 0, 0)
  Q_PLUGIN_METADATA(IID "org_mitk_gui_qt_igttracking")
#endif
  Q_INTERFACES(ctkPluginActivator)

public:

  void start(ctkPluginContext* context) override;
  void stop(ctkPluginContext* context) override;

  static ctkPluginContext* GetContext();

private:
  static ctkPluginContext* m_Context;

};
```

```
ctkPluginContext* PluginActivator::m_Context = nullptr;

void PluginActivator::start(ctkPluginContext* context)
{
  m_Context = context;

      …

}

void PluginActivator::stop(ctkPluginContext* context)
{
  m_Context = nullptr;

  Q_UNUSED(context)
}

ctkPluginContext* PluginActivator::GetContext()
{
  return m_Context;
}
```

How to use a MS in a Plugin – part 2


```
#include "mitkPluginActivator.h"

ctkPluginContext* pluginContext = mitk::PluginActivator::GetContext();

QString interfaceName =
       QString::fromStdString(us_service_interface_iid<mitk::TrackingDeviceTypeCollection>());

QList<ctkServiceReference> serviceReferences =
       pluginContext->getServiceReferences(interfaceName);

const ctkServiceReference& serviceReference = serviceReferences.at(0);

m_DeviceTypeCollection =
       pluginContext->getService<mitk::TrackingDeviceTypeCollection>(serviceReference);

deviceTypeCollection->RegisterTrackingDeviceType(
                                    new mitk::VirtualTrackerTypeInformationMBI());

                                        pluginContext->ungetService(serviceReference);
```



Additional hints

- AUTOLOAD_WITH
  - Only load non-Qt dependent modules with core
  - Use AUTOLOAD_WITH <MyModule> instead
  - Keep Load() function as slim as possible (no Qt initialization!)
  - Be careful with static variable initialization during autoloading

Take home message

- How to write a Microservice
  1. Create own Module for autoloading
     a) `AUTOLOAD_WITH …`
  2. Write Microservice
     a) `MITK_DECLARE_SERVICE_INTERFACE`
     b) `context->RegisterService`
  3. Write autoload classes
     a) `us::ModuleActivator`
     b) `US_EXPORT_MODULE_ACTIVATOR`

- How to use a Microservice
  - Module Context
  - Plugin Context

References

- http://docs.mitk.org/nightly/MicroServices_Overview.html
- http://docs.mitk.org/nightly/MicroServices_AutoLoading.html

- Also consider Doxygen documentation of the Microservice framework!


Auf Wiedersehen
im DKFZ!
dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION
50 Years – Research for
A Life Without Cancer