C++ Microservices Pt. I

Declearing Services

Intro

- Dynamic Service Registry
- based on service layer specified in OSGi R4.2 specs

Target Applications

- Relevant for large TechStacks
- Multiple Modules
- Multiple Plugins
- Configurability makes assumptions difficult

Basic Idea

- Some Classes have Service Role
  - Unnecessary Instantiantion → Resources!
- Some have Singleton Nature
  - Singular Instantiation necessary

Service Registry

- Idea: Make available Registry
- Consuming code refers to registry

Devices in MITK

- Devices ↔ Services?
- Hardware access conflicts
- Configuration Workflow

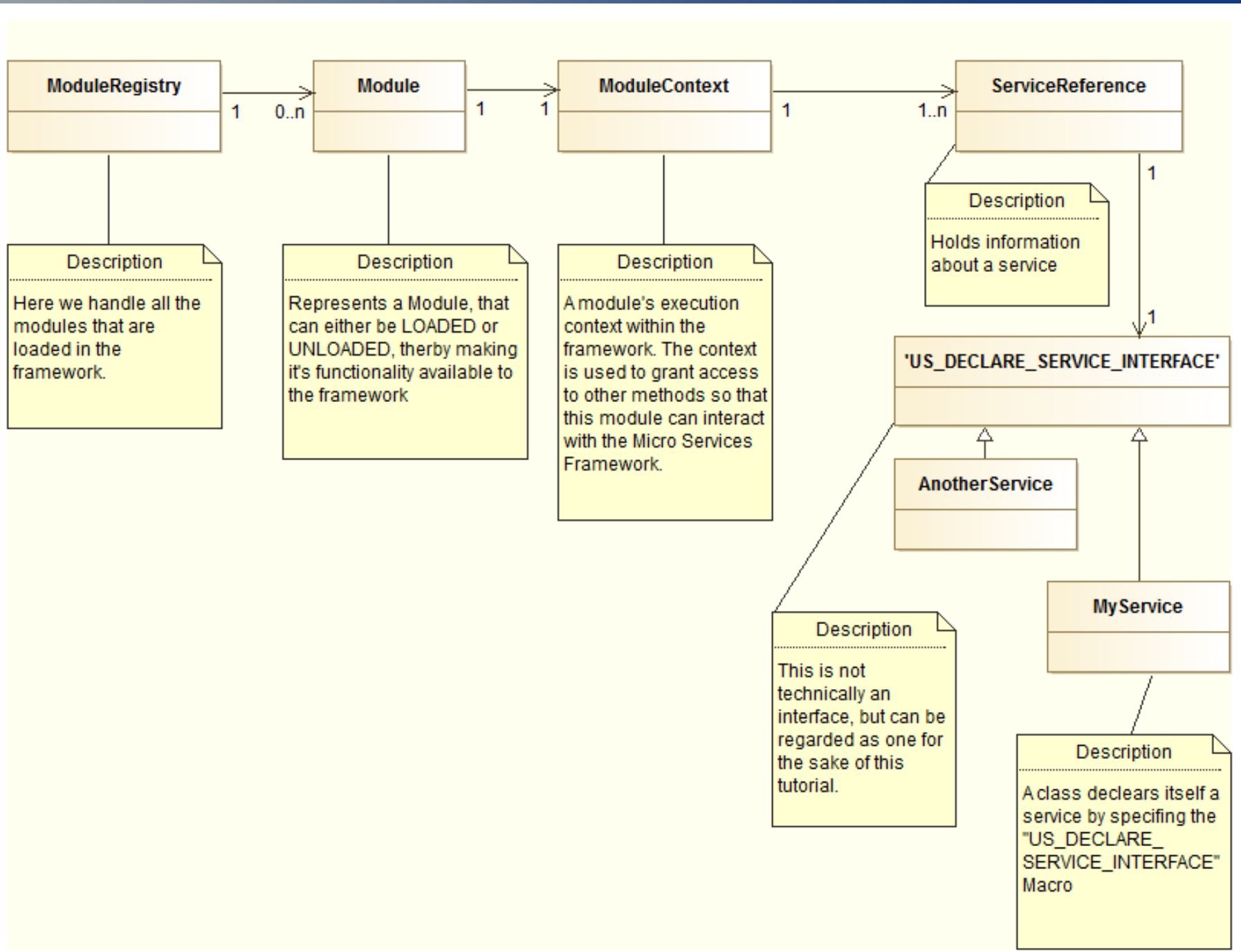
Example

- Consider: new MBI-Plugin „UltraReg“ needs Devices to implement new Technique / Algorithm
  - Requires Tracking Devices
  - Requires Imaging Device
- MITK-IGT
  - Registers Connected Tracking Devices
- MITK-TOF
  - Registers TOF Devices

Example

- UltraReg only needs to query Service Registry for devices
- Does not need any connection logic

Microservice Architecture



3 Steps To Microservices

- In the following:
  - Three simple steps to Microservice glory!

1) Make the class a service

- make a class declare itself a Service Interface

```
#include <usServiceInterface.h>


US_DECLARE_SERVICE_INTERFACE(DictionaryService,
    "org.mitk.services.MyService")
```

- By convention put this line before the last #endif
- The chosen name will be used to refer to this service in the future

2) Register the Service

```
// get the module context
mitk::ModuleContext* context = GetModuleContext();

// define service properties
ServiceProperties props;

// Fill Properties
props["MyServicePropert"] =  “MyValue”

m_ServiceRegistration = context->RegisterService<mitk::USDevice>(this,
     props);

return true;
```

3) Access Services Everywhere

```
#include <usGetModuleContext.h>  // Don‘t forget this include!

mitk::Module* mitkUS =
    mitk::ModuleRegistry::GetModule("MyModuleName");

m_MitkUSContext = mitkUS->GetModuleContext();

mitk::ServiceReference serviceRef =
   m_MitkUSContext->GetServiceReferences<mitk::USDevice>();

// Convert Service References to your service class
std::list<mitk::ServiceReference>::const_iterator iterator;

mitk::MyService::Pointer myService =
    m_MitkUSContext->GetService<mitk::MyService>(serviceRef);
```

Example Implementation

- For sample implementations, refer to mitk::USDevice

QmitkServiceListWidget

- Module QMITK: QmitkServiceListWidget.cpp
- Offers lots of abstraction for working with Microservices on an User Interface Level
- Distinct Speech (German) for this: Microservices Pt. II