Fantastic (DICOM) meta information

and where to find|put it

Bug squashing seminar; 2016-11-25

Ralf Floca

Target groups

1. You need DICOM meta information of loaded data within MITK?
2. You generate data and want to attach meta information to it?

You are in!

How to get DICOM meta information loaded? (1/2)

- Easiest way: Use mitk::IDICOMTagsOfInterest micro service

```
/** \brief Add an tag to the TOI.
 * If the tag was already added it will be overwritten with the passed values.
 * \param[in] tag Tag that should be added.
 * \param[in] makePersistant Indicates if the tag should be made persistant if possible
    via the IPropertyPersistence service.*/
 virtual void AddTagOfInterest(const mitk::DICOMTagPath& tag, bool makePersistant = true) = 0;
```

Example

```
auto tagsOfInterestService = SomeFunctionToGetTheService();

//For simple („1st level“) tags like DICOM modality
tagsOfInterestService->AddTagOfInterest(DICOMTagPath(0x0008, 0x0080));
tagsOfInterestService->AddTagOfInterest(DICOMTag(0x0008, 0x0080)); //alternative for simple tags

//For nested tags (e.g. (sub) items in a DICOM sequence) like the DICOM total dose of radionuclides
mitk::DICOMTagPath radionuclideTotalDoseTag;
radionuclideTotalDoseTag.AddAnySelection(0x0054, 0x0016).AddElement(0x0018, 0x1074);

tagsOfInterestService->AddTagOfInterest(radionuclideTotalDoseTag);
```

How to get DICOM meta information loaded? (2/2)

- Now the tags get extracted when you use the mitk IO
- Currently supported data
  - DICOM Images (via DICOM Editor and normal IO)
  - DICOM RT Dose / DICOM RT StructureSet
  - DICOM Segmentation
- Nice benefit: the information also gets persisted when you
  - save the scene
  - save images/doses/segmentations in ITK formats (that support meta information serialization; e.g. NRRD, meta image)

And where to find it? (1/4)

- Stored as properties of the data instance.
- Property naming convention:
  - Starts with “DICOM.”
  - Tags are uppercase hex: “<group number>.<element number>”
  - Item selections are indicated by: “[<index>]”
  - Elements are separated by “.”
- Example:
  - Modality: DICOM.0008.0080
  - Radionuclide Total Dose (first item): DICOM.0054.0016.[0].0018.1074

And where to find it? (2/4)

- Easiest way to regard the naming conventions
- For simple cases (where you only want MitkCore dependency):
  In mitkPropertyNameHelper.h

```
/** returns the correct property name for a simple DICOM tag. The tag is defined
 * by the passed group and element number (both in hex). */
std::string mitk::GeneratePropertyNameForDICOMTag(unsigned int group, unsigned int element);
```

- For (complex) cases:
  In mitkDICOMTagPath.h (Module MitkDICOMReader)

```
/** Converts the passed property name into a tag path. If the property name cannot be converted
 into a valid path, the returned path is empty.*/
mitk::DICOMTagPath mitk::PropertyNameToDICOMTagPath(const std::string& propertyName);

/** returns the correct property name for a given DICOMTagPath instance. */
std::string mitk::DICOMTagPathToPropertyName(const mitk:: DICOMTagPath& tagPath);
```

And where to find it? (3/4)

- Extracted DICOM information is stored as:

  mitk::TemperoSpatialStringProperty

- Access it as a simple string property (like in the past) via

```
/**Returns the value of the first time point in the first slice.
 * If now value is set it returns an empty string.*/
ValueType mitk::TemperoSpatialStringProperty::GetValue() const;
virtual std::string mitk::TemperoSpatialStringProperty::GetValueAsString() const override;
```

And where to find it? (4/4)

- Advanced access (temporal and spatial resolved) via

```
/**Returns the value of the passed time step and slice. If it does not exist
 and allowedClosed is true it will look for the closest value. If nothing could
 be found an empty string will be returned.*/
ValueType mitk::TemperoSpatialStringProperty::GetValue(const TimeStepType& timeStep,
                                                       const IndexValueType& zSlice,
                                                       bool allowCloseTime = false,
                                                       bool allowCloseSlice = false) const;

ValueType mitk::TemperoSpatialStringProperty::GetValueBySlice(const IndexValueType& zSlice,
                                                              bool allowclosed = false) const;

ValueType mitk::TemperoSpatialStringProperty::GetValueByTimeStep(const TimeStepType& timeStep,
                                                                 bool allowclosed = false) const;
```

- An example using the features: “DICOM Tag Inspector” plugin

Generating own meta data…

You should regard the following good Practice:

- Don’t reinvent the wheel (aka: your own naming convention)
- Use DICOM tags to store the information!!!
- Use the introduced functions to generate compliant property names

  `GeneratePropertyNameForDICOMTag` and `DICOMTagPathToPropertyName`

Why regarding the policy… Typical thoughts + answers

“Why? I don’t even generate DICOM objects.”

- Doesn’t matter, you can nevertheless profit
- DICOM defines standard naming for a lot of information
- Use it properly and interpretation is independent from you code! ☺

Why regarding the policy… Typical thoughts + answers

“But I don’t know the right tag?”

- RTFM ;)
- Or just ask. We can help to check for a fitting tag (and correct interpretation)

Why regarding the policy… Typical thoughts + answers

“But I don’t want to store it as DICOM?”

- No show stopper. You profit from the mentioned benefits and store it in other suitable formats.
- E.g. NRRD or meta images are already supported

Why regarding the policy… Typical thoughts + answers

“But it the naming is ugly as hell and not human readable!”

- Use the PropertyDescriptionService
- The DICOMTagOfInterest service stores a human readable description automatically for you.

What's next…

- But how to define own persistency naming strategies?
- But how to use it with own property types?
- …

To be continued…