BVM-Tutorial 2010: Entwicklung interaktiver medizinischer Bildverarbeitungssysteme mit MITK

Daniel Maleike, Michael Müller, Alexander Seitel,
Marco Nolden

dkfz. DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

Ablauf des Tutorials

- Einführung in ITK, VTK und MITK

Praxis 1:
Demo der Qt4 Basisapplikation „ExtApp“
Download und Einrichtung der MITK Umgebung

- MITK Grundkonzepte

Kaffeepause

- Grundlagen des BlueBerry Framework

Praxis 2:
Erstellung eines Applikationsmoduls
Interaktion und Anbindung von Algorithmen

- Image Guided Therapy mit MITK: Konzepte und Demonstration
- Ausblick und Ende

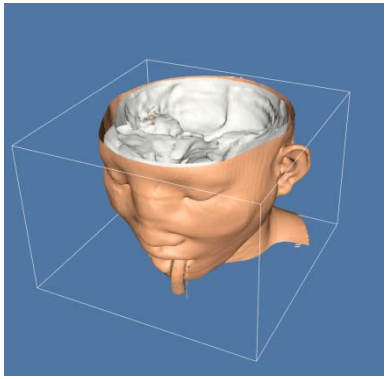
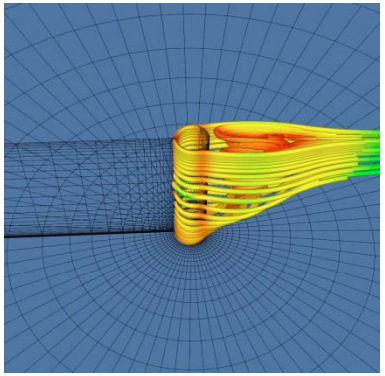
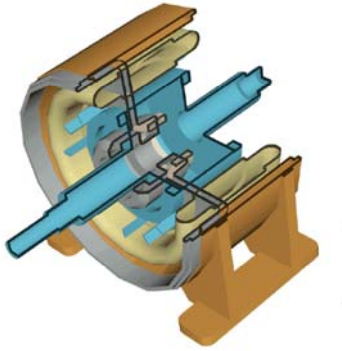
Fragen an die Teilnehmer

- Erfahrung mit ITK, VTK, MITK?
- Was ist das Hauptinteresse am MITK?
  - Plattform für End-User Applikationen
  - Prototyp-Plattform für Algorithmen-Entwicklung
  - Navigation
  - Sonstiges

VTK, ITK, MITK

VTK

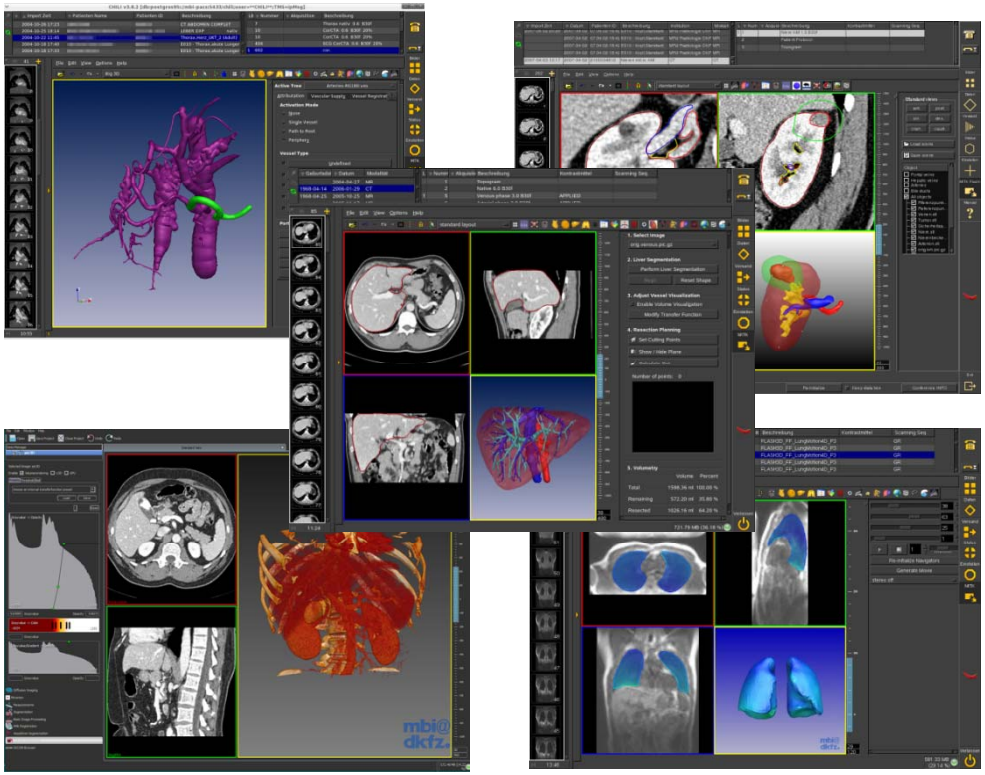
- Visualization Toolkit
- *The Visualization Toolkit An Object-Oriented Approach to 3D Graphics*. Will Schroeder, Ken Martin, and Bill Lorensen (1993)
- Unterstützung durch GE Research
- 1998: Gründung von Kitware Inc.
- Implementiert in C++, Anbindung von TCL, Python, Java …
- Viele Visualisierungsverfahren, einfache Bildverarbeitungs- und GUI Komponenten

ITK

- Insight Segmentation and Registration Toolkit
- Ausschreibung durch NLM und NIH 1999
- Sechs Partner aus Industrie und Wissenschaft: GE, Kitware, University of North Carolina …
- Datenstrukturen und Algorithmen für Registrierung und Segmentierung in medizinischen Anwendungen
- Keine GUI oder Applikationskomponenten
- Generische Programmierung, “advanced” C++
- Open Source Prozess, Insight Journal

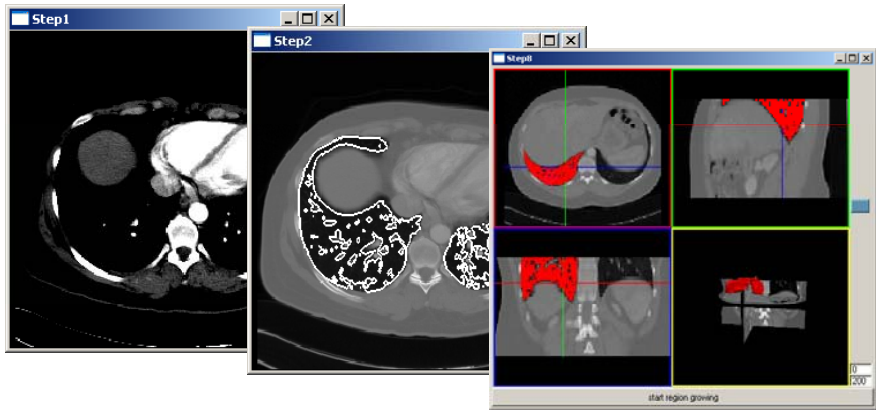
MITK Softwareplattform des DKFZ



- Software-Bibliothek,
  seit 2002 vom DKFZ entwickelt
- Basis für sämtliche Anwendungen
  der Abteilung
- Ca. 500.000 Zeilen Open Source
- Etabliert in Forschung u. Bildung
  (ca. 200 forschende Anwender)

Open Source MITK

MITK kann auf verschiedene Weise verwendet werden, man kann:

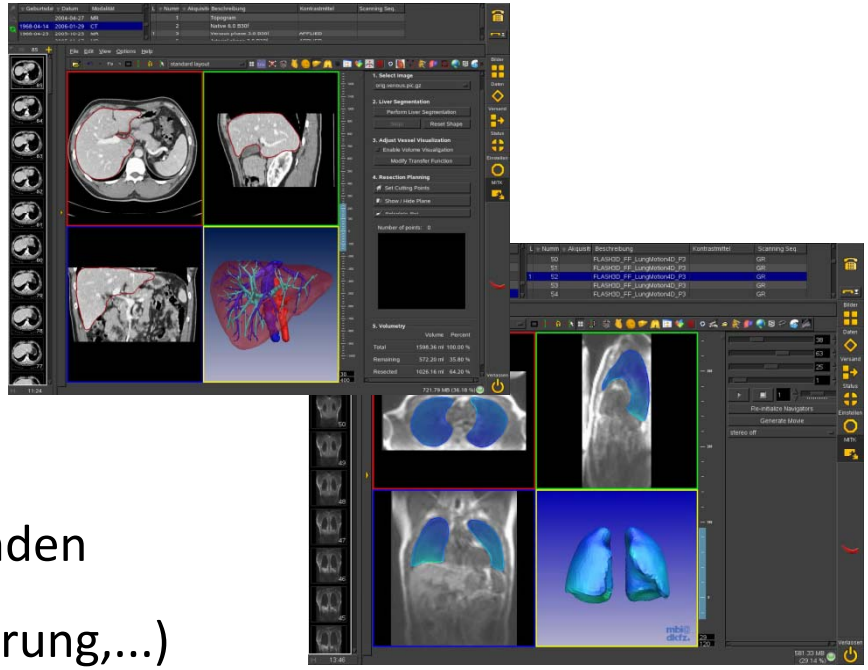


1. Neue Anwendungen von Beginn an entwickeln

   → MITK wird als **SW-Bibliothek** verwendet

2. Die Open-Source MITK-Applikation erweitern (neue Module, Plug-Ins,...)

   → MITK wird als **Application-Framework** verwendet



3. Die existierende **3M3-Applikation** verwenden (Seg., Reg., Vermessung, 3D (+t) Visualisierung,...)

Open-source Toolkits

Powerful toolkits for

- Visualization: VTK (http://www.vtk.org)
- Segmentation / Registration: ITK (http://www.itk.org)

But:

insufficient support for
interactive, multi-view software

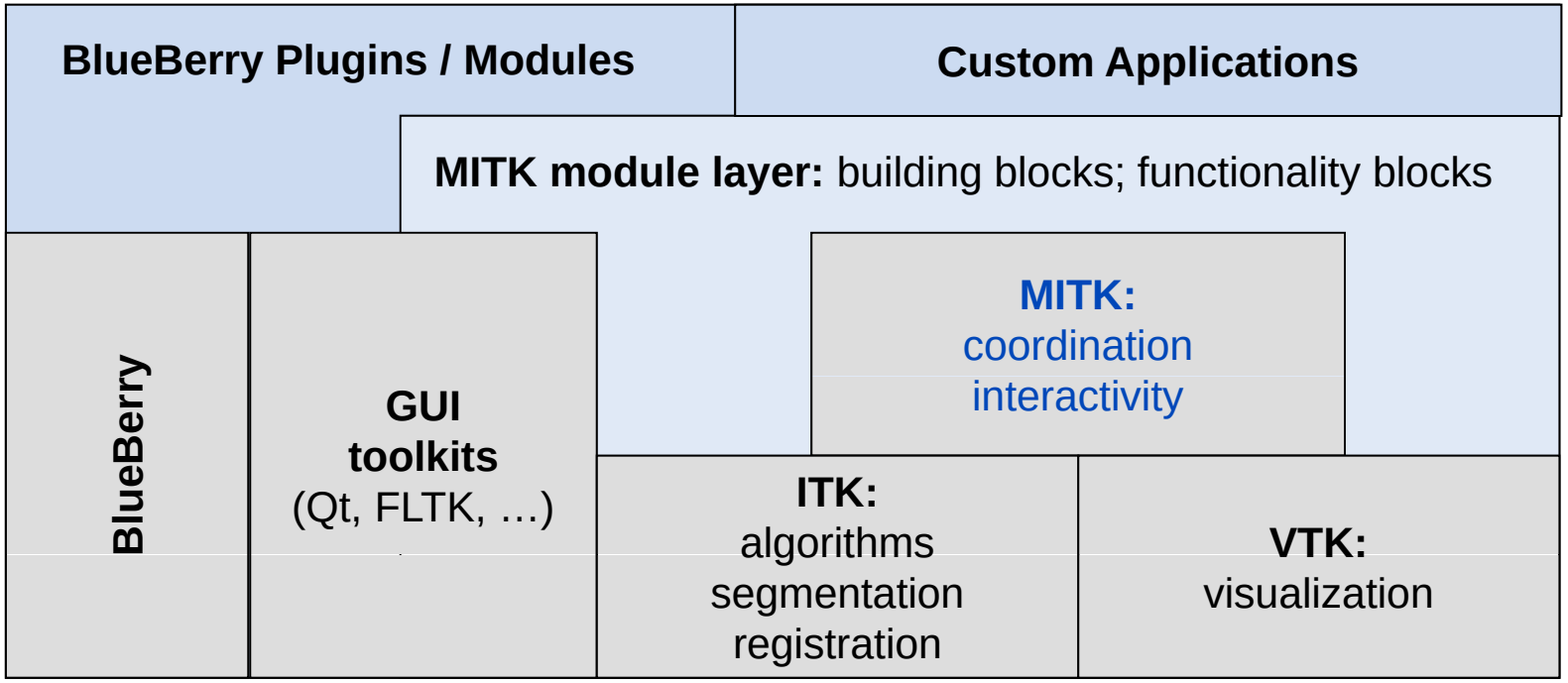
MITK …

- uses parts of NA-MIC: **ITK** & **VTK**
- adds features outside the scope of boths

➔ is not at all a competitor to VTK or ITK

Medical Imaging Interaction Toolkit (MITK)

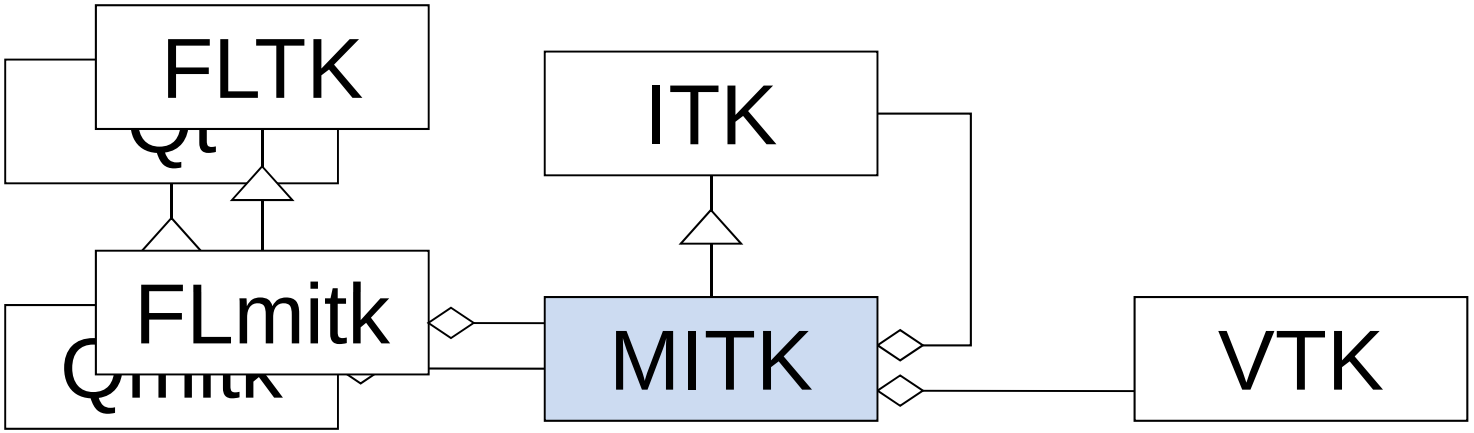
- open-source C++ toolkit based on ITK/VTK
- coordination of visualizations and interactions
- combine modules developed independently from each other



MITK

- Object-oriented C++ framework / toolkit
- BSD-style license, almost identical to VTK / ITK
- Supports
  - Linux, Windows, Mac OSX
  - gcc 4.x, VC8, VC9
  - Latest VTK release
  - Latest two ITK releases
- MITK-core does not depend on a GUI toolkit
- MITK-application-level provides
  - Qt4 base application
  - Many Qt widgets
  - FLTK example
  - wxWidgets port (part of GIMIAS)

Re-use of design and concepts



- MITK‘s core is GUI independent

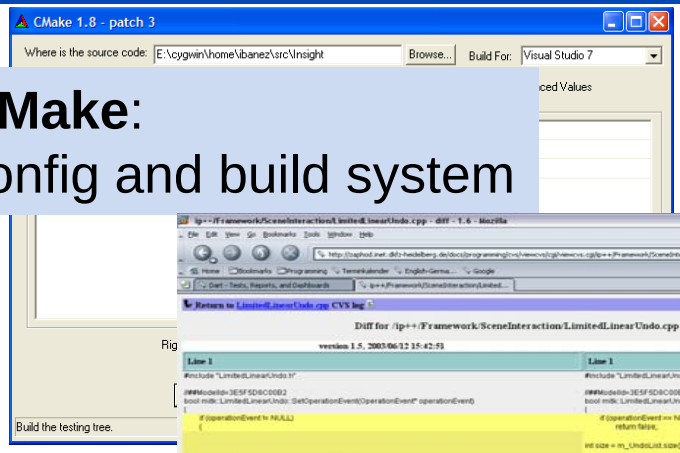
Open source MITK statistics

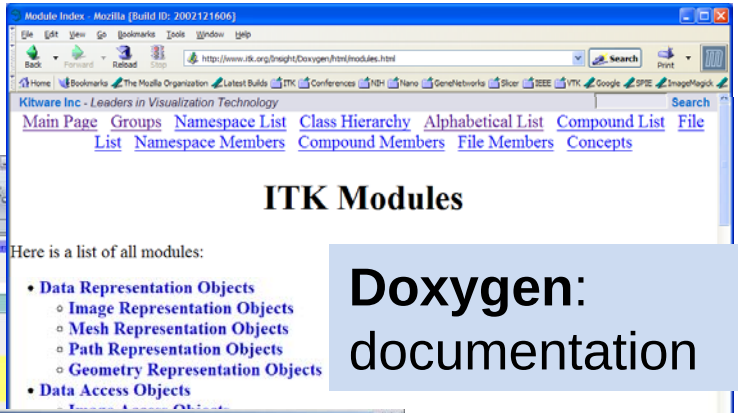
- ~150 package downloads / month
- ~100 developers use the public read-only repository
- Independent research projects based on the MITK platform
  - MIT Boston
  - Mayo Clinic Rochester
  - RWTH Aachen
  - Fraunhofer IGD Darmstadt
  - Charité Berlin
  - Pompeu Fabra University, Barcelona
  - ...

MITK uses NA-MIC tools and software process

dkfz.

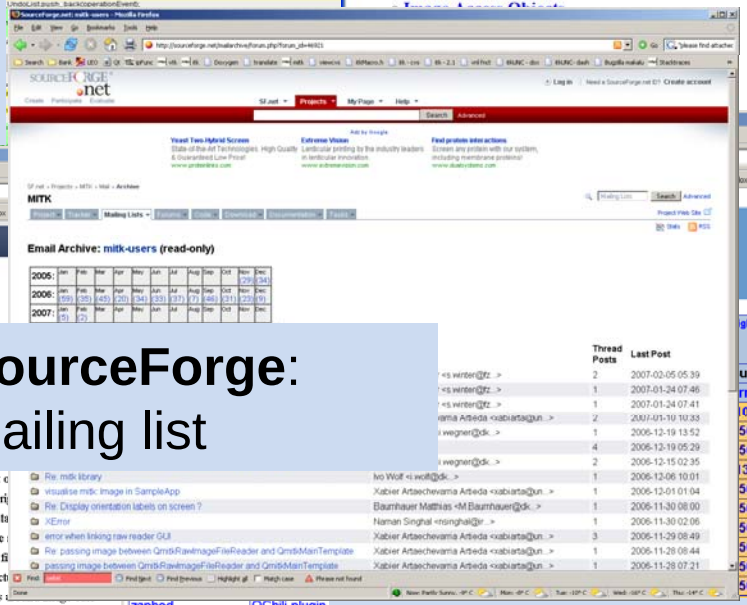
CMake:
config and build system



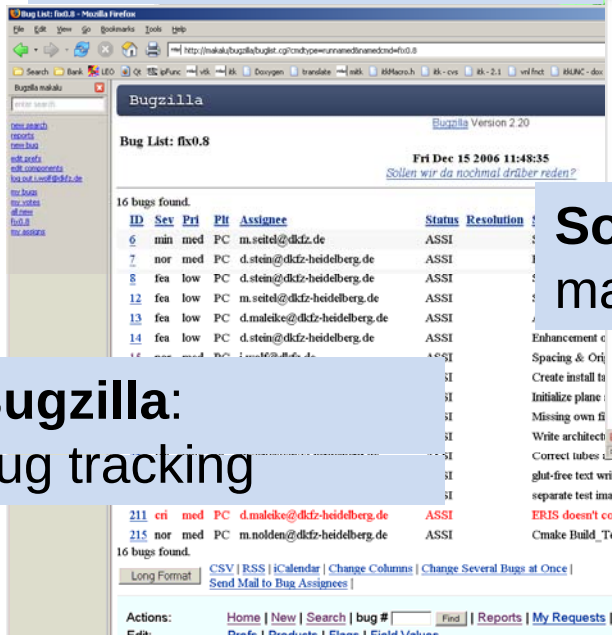


Doxygen:
documentation

Subversion:
version management



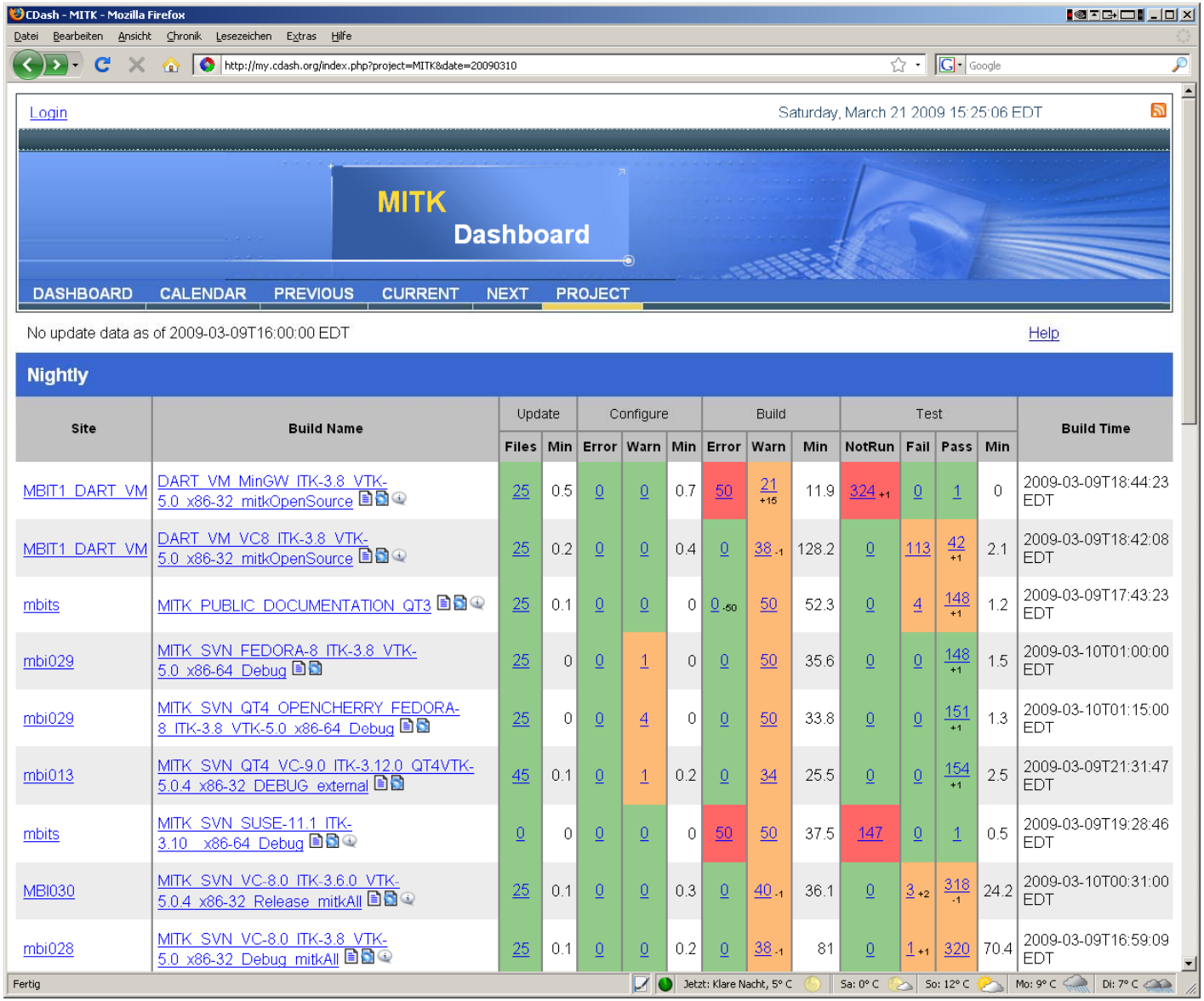
SourceForge:
mailing list



Bugzilla:
bug tracking

CTest/CDash:
automatic builds and test runs

Continuous Integration with public dashboard



Nightly

Site	Build Name	Update		Configure			Build			Test				Build Time
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min	
MBIT1_DART_VM	DART_VM_MinGW_ITK-3.8_VTK-5.0_x86-32_mitkOpenSource	25	0.5	0	0	0.7	50	21 +15	11.9	324 +1	0	1	0	2009-03-09T18:44:23 EDT
MBIT1_DART_VM	DART_VM_VC8_ITK-3.8_VTK-5.0_x86-32_mitkOpenSource	25	0.2	0	0	0.4	0	38 -1	128.2	0	113	42 +1	2.1	2009-03-09T18:42:08 EDT
mbits	MITK_PUBLIC_DOCUMENTATION_QT3	25	0.1	0	0	0	0 -50	50	52.3	0	4	148 +1	1.2	2009-03-09T17:43:23 EDT
mbi029	MITK_SVN_FEDORA-8_ITK-3.8_VTK-5.0_x86-64_Debug	25	0	0	1	0	0	50	35.6	0	0	148 +1	1.5	2009-03-10T01:00:00 EDT
mbi029	MITK_SVN_QT4_OPENCHERRY_FEDORA-8_ITK-3.8_VTK-5.0_x86-64_Debug	25	0	0	4	0	0	50	33.8	0	0	151 +1	1.3	2009-03-10T01:15:00 EDT
mbi013	MITK_SVN_QT4_VC-9.0_ITK-3.12.0_QT4VTK-5.0.4_x86-32_DEBUG_external	45	0.1	0	1	0.2	0	34	25.5	0	0	154 +1	2.5	2009-03-09T21:31:47 EDT
mbits	MITK_SVN_SUSE-11.1_ITK-3.10_x86-64_Debug	0	0	0	0	0	50	50	37.5	147	0	1	0.5	2009-03-09T19:28:46 EDT
MBI030	MITK_SVN_VC-8.0_ITK-3.6.0_VTK-5.0.4_x86-32_Release_mitkAll	25	0.1	0	0	0.3	0	40 -1	36.1	0	3 +2	318 -1	24.2	2009-03-10T00:31:00 EDT
mbi028	MITK_SVN_VC-8.0_ITK-3.8_VTK-5.0_x86-32_Debug_mitkAll	25	0.1	0	0	0.2	0	38 -1	81	0	1 +1	320	70.4	2009-03-09T16:59:09 EDT

Quality Management

ZERTIFIKAT ◆ CERTIFICATE ◆ 認証書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

TÜV SÜD Product Service

ZERTIFIKAT

Nr. Q1N 05 12 57646 001

Zertifikatsinhaber: **Deutsches Krebsforschungszentrum**
Abt. Medizinische und
Biologische Informatik
Im Neuenheimer Feld 280
69120 Heidelberg
DEUTSCHLAND

dkfz. DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

Betriebsstätte(n): Deutsches Krebsforschungszentrum Abt. Medizinische und Biologische Informatik
Im Neuenheimer Feld 280, 69120 Heidelberg, DEUTSCHLAND

Zertifizierungs-zeichen: TÜV SÜD ISO 13485:2003

Geltungsbereich: **Design, Entwicklung und Produktion von medizinischer Bildverarbeitungssoftware zur Diagnose- und Therapieunterstützung**

Angewandte Norm(en): ISO 13485:2003
Medizinprodukte - Qualitätsmanagementsysteme - Anforderungen für regulatorische Zwecke
Medical Devices - Quality Management Systems - Requirements for regulatory purposes

Die Zertifizierstelle von TÜV Product Service GmbH bescheinigt, dass das oben genannte Unternehmen ein Qualitätsmanagement-System eingeführt hat und anwendet, das den Anforderungen der genannten Norm(en) entspricht. Umseitige Hinweise sind zu beachten.

Bericht Nr.: 70108974
Gültig bis: 2008-12-31

Datum, 2005-12-28

Reiner Krumme

TÜV SÜD 254082

Seite 1 von 1

TÜV Product Service GmbH
TÜV SÜD Gruppe - Zertifizierstelle
Ridlerstr. 65 - 80339 München
Germany

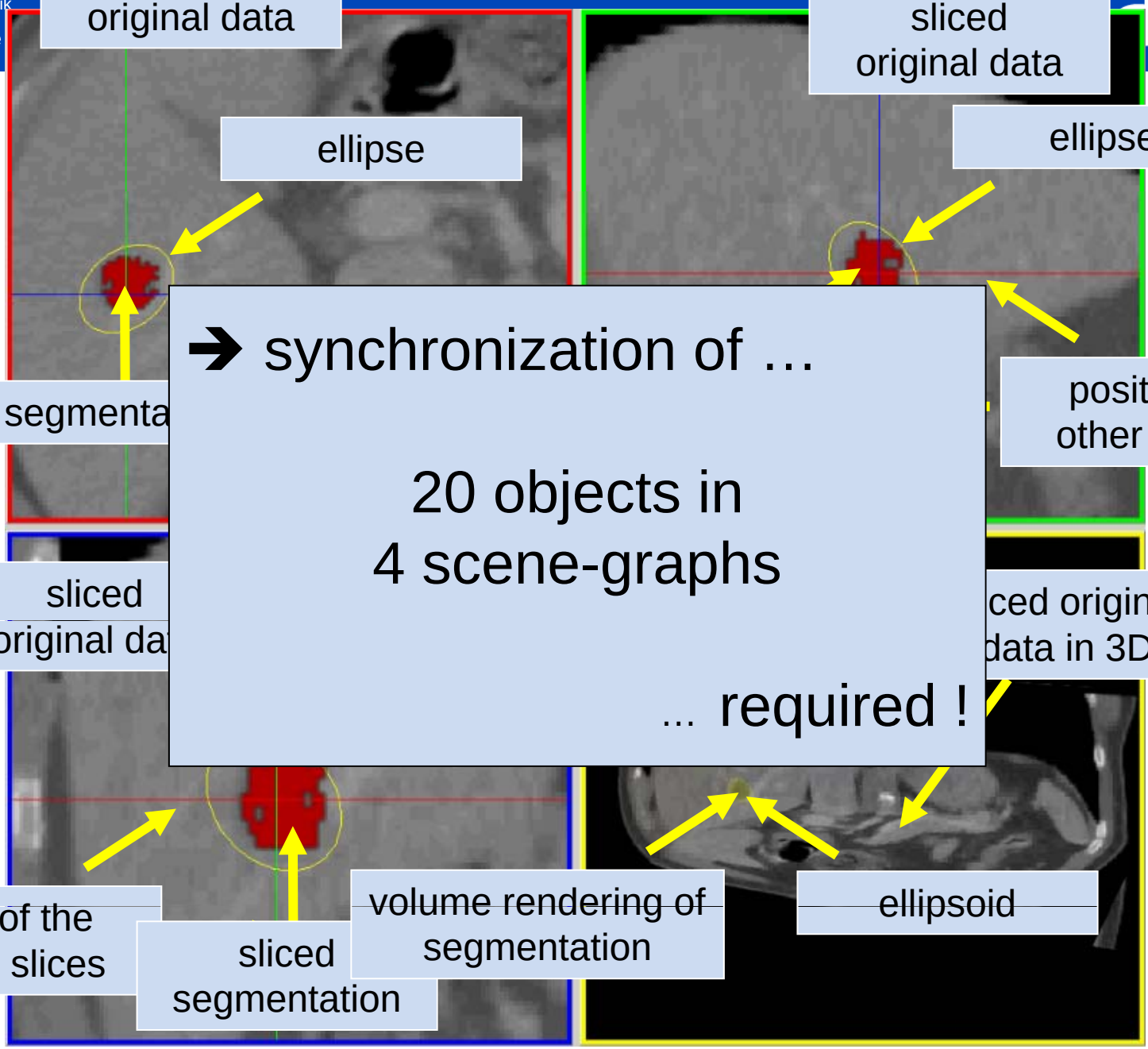
ZLG **Akkreditiert durch** Zentralstelle der Länder für Gesundheitsschutz bei Arzneimitteln und Medizinprodukten
ZLG-ZQ-999.98.12-46

- QM-System according to DIN EN ISO 13485
- Working group „Clinical Applications“
- Support and application development for MITK

mint medical

What MITK does –
a quick overview

DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

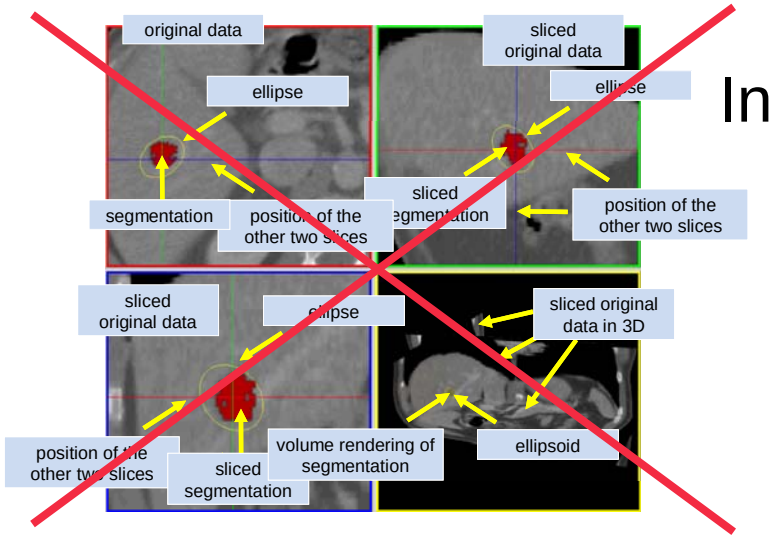


➔ synchronization of …

20 objects in
4 scene-graphs

… required !

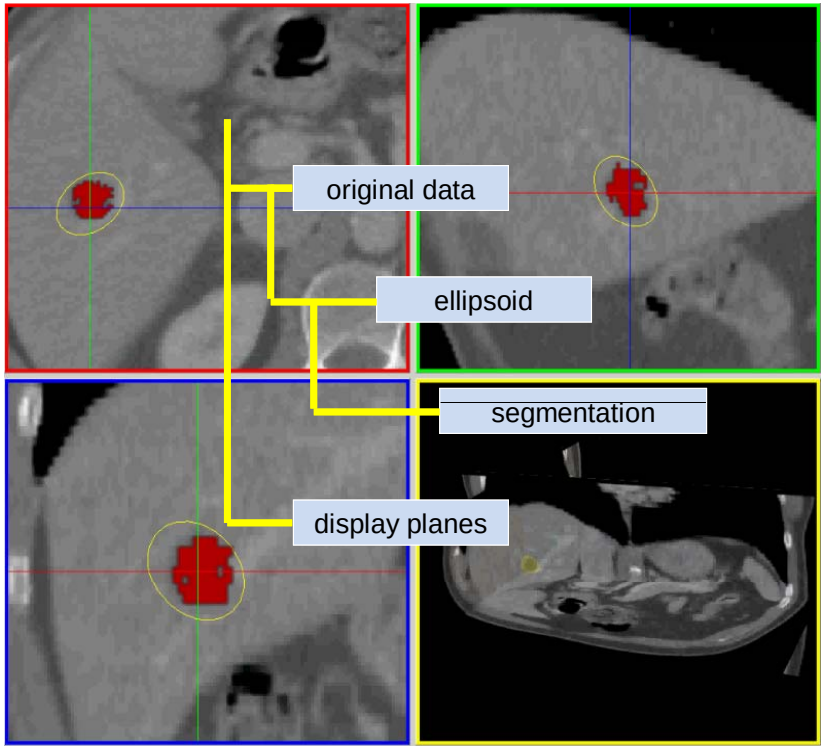
Getting out of the maze …



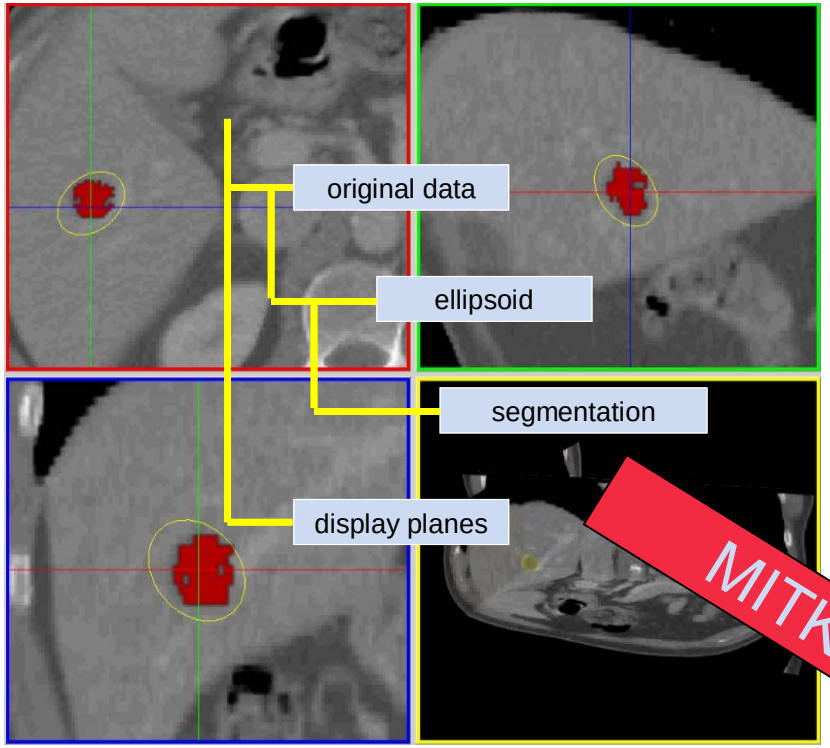
Instead of creating **many** scene-graphs with **even more** elements …

… create a **single data repository**

with a **few data-objects**!



MITK: Data repository instead of scene-graphs

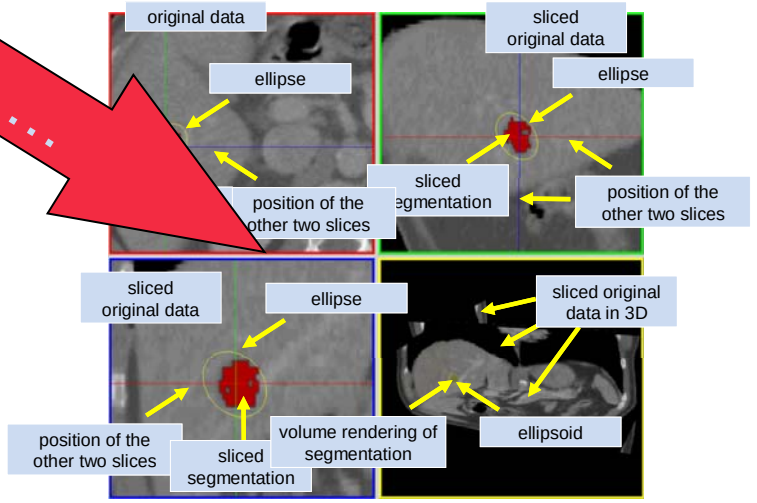


MITK takes the data repository …

and builds …

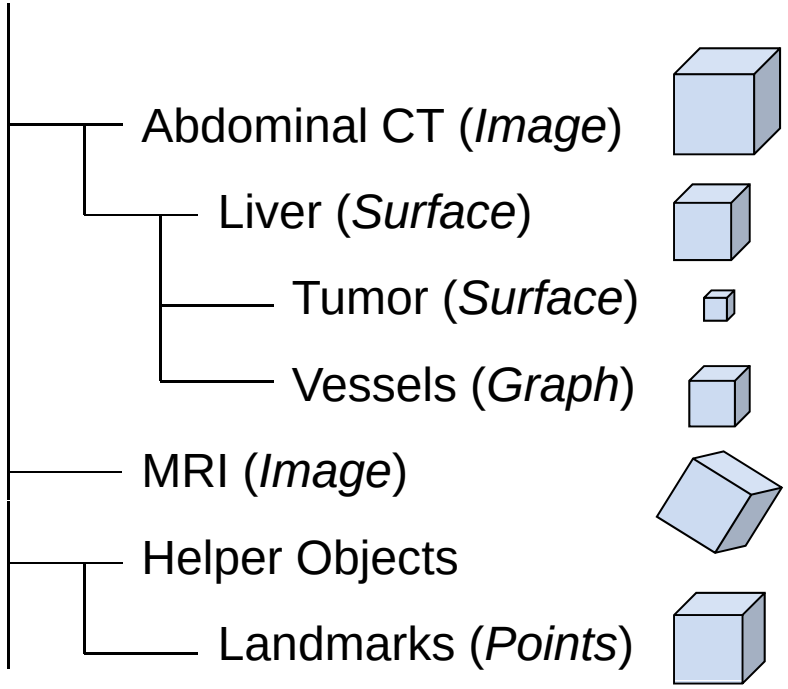
➔ VTK scene graphs

MITK creates …

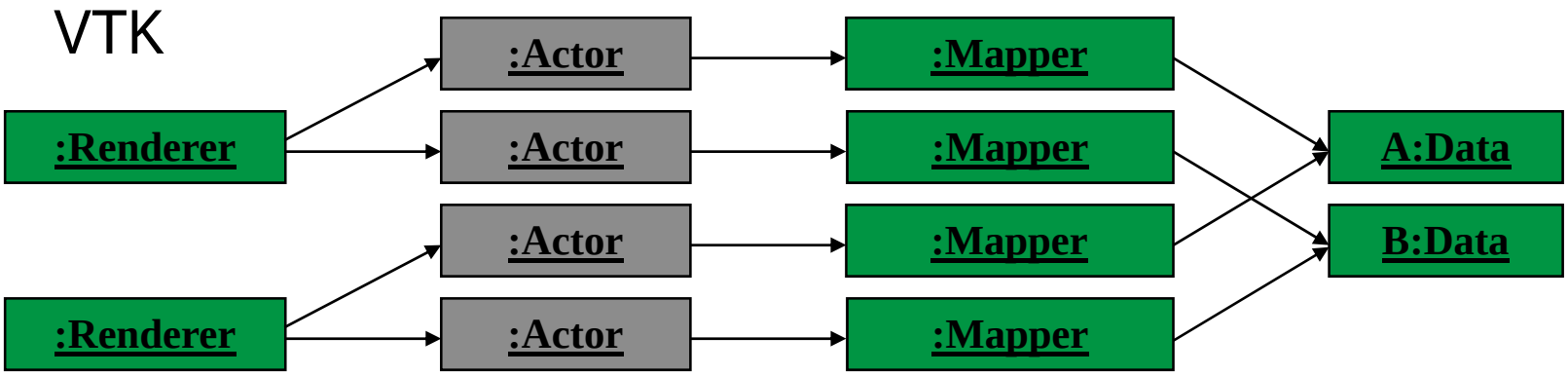


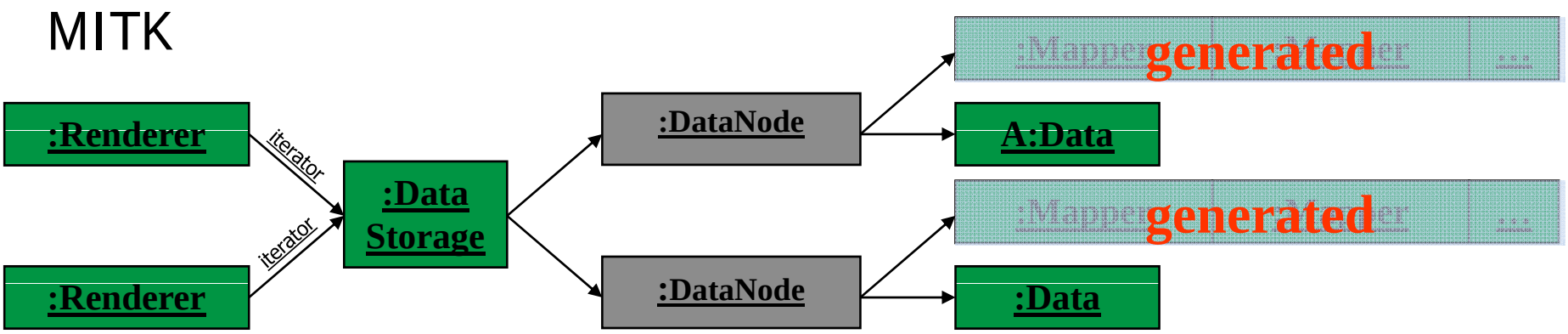
Data repository

- Repositories for sharing data objects between modules
- Any number of data objects
- Any kind of data objects
- Data objects with geometry frame (bounding-box, transform, etc.)

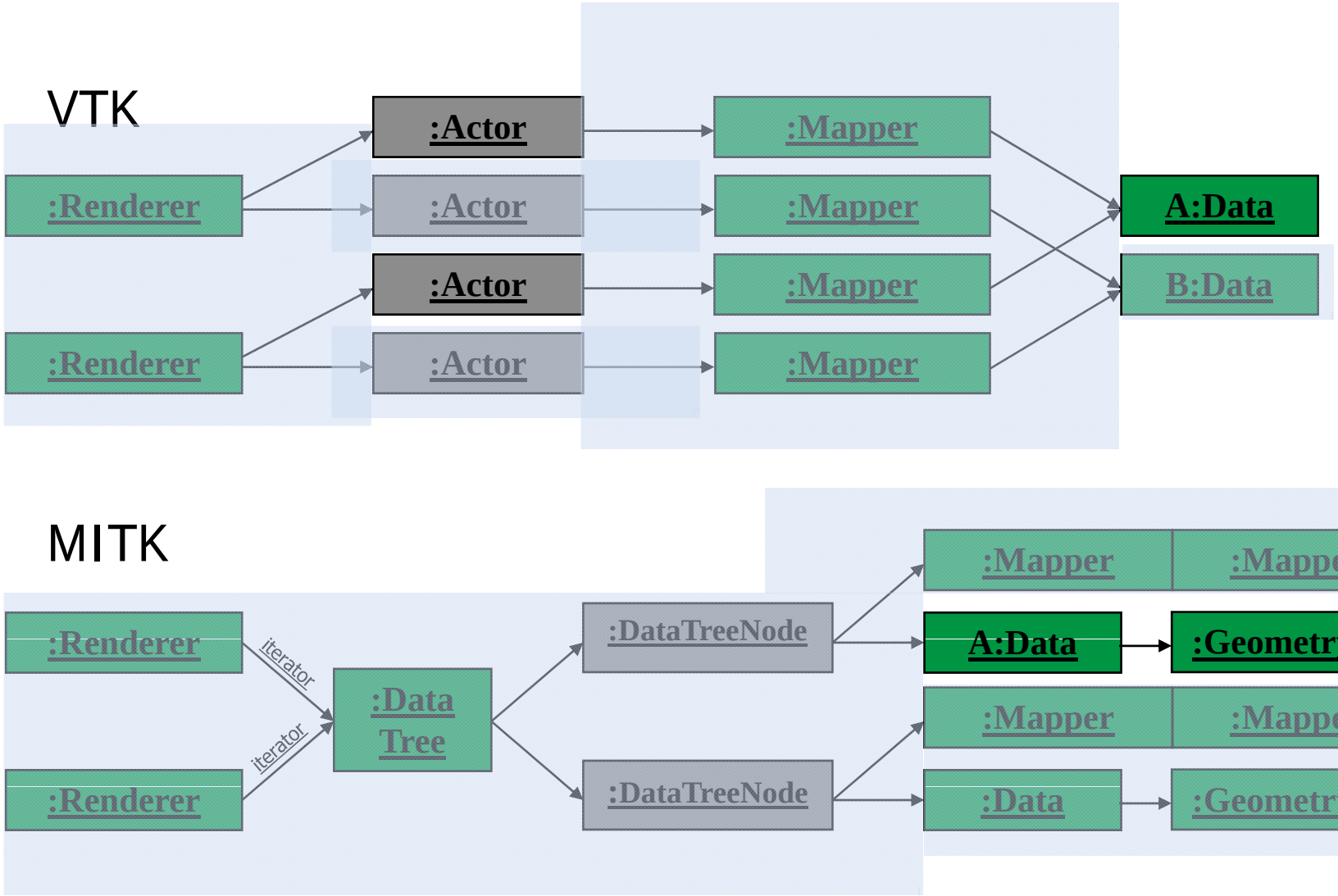


Rendering VTK vs. MITK

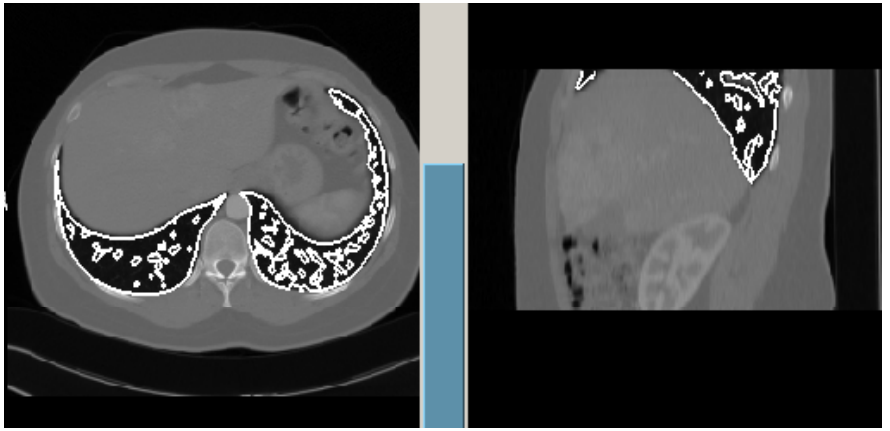




Positioning of an object



Rendering the data repository



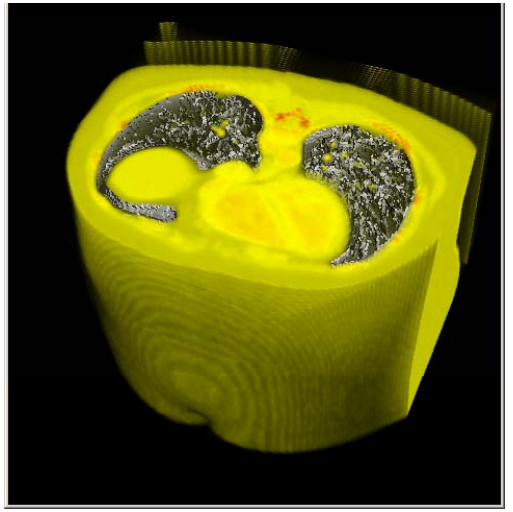
Render windows:

- **single** RenderWindow class
- **different types** of views
  - ➔ 2D/3D
  - ➔ special views definable (e.g., for AR)

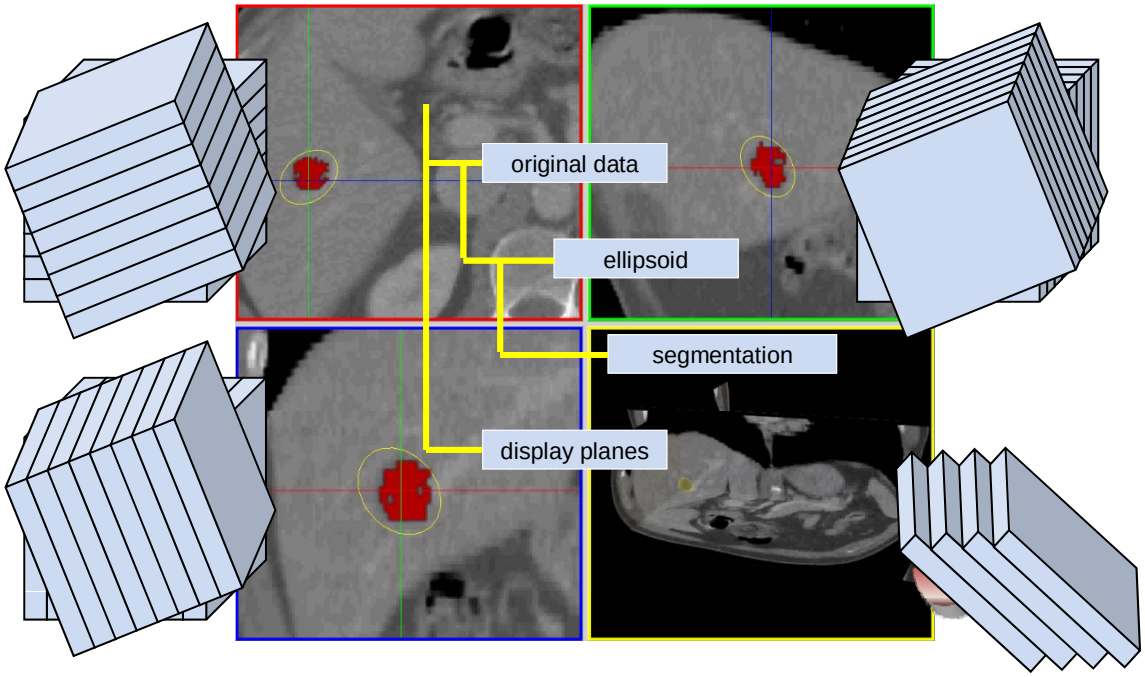
```
renderer->SetMapperID(BaseRenderer::Standard3D);
```

- **point** to the **data repository**
  - ➔ **any number of views** on the data:

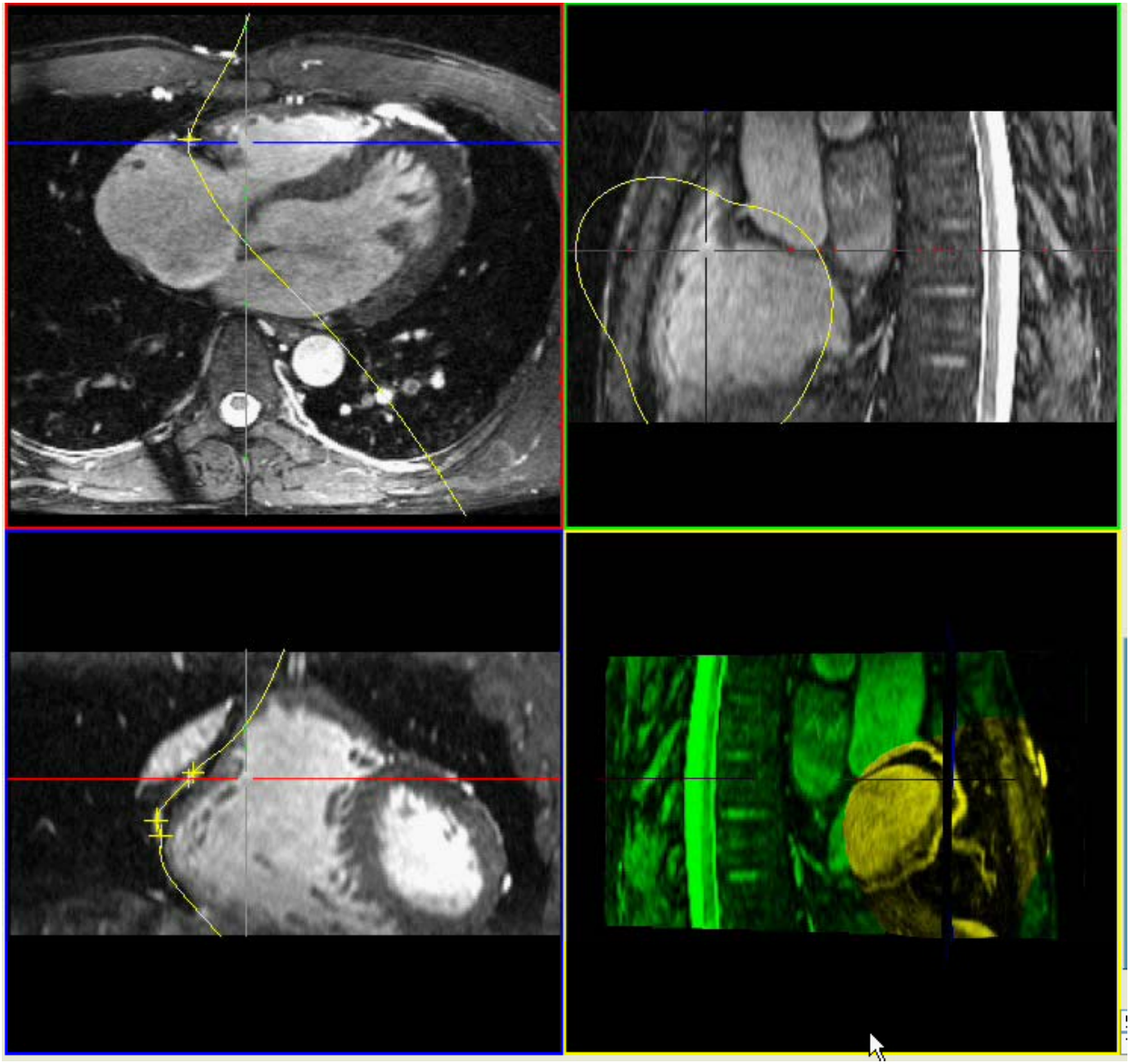
```
renderer1->SetData(repository);
renderer2->SetData(repository);
...
```



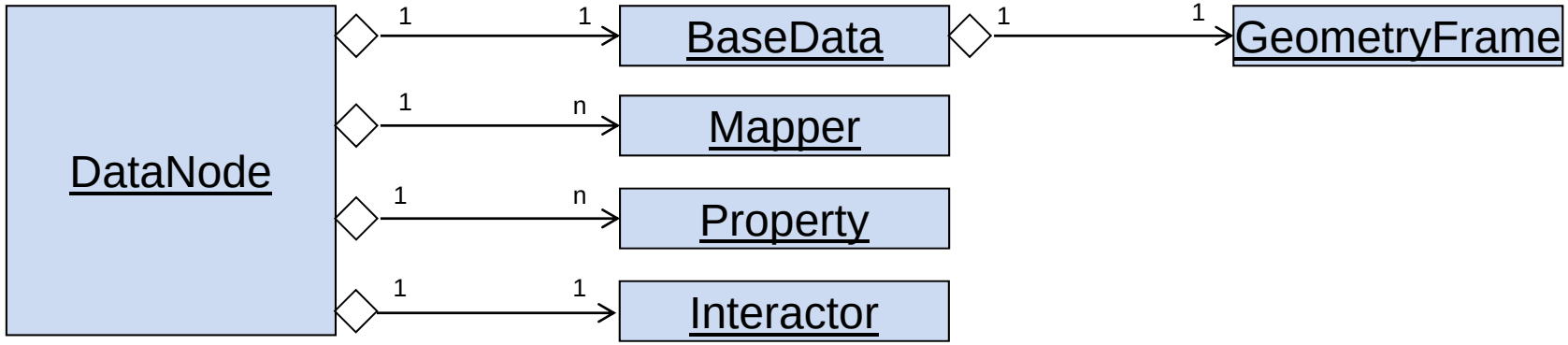
Defining
how we want to see the data …



Render and interact on curved planes



The nodes in the data repository



BaseData: the actual data: images, surfaces, etc.

GeometryFrame: position and orientation in space

Mappers: render the data into a renderwindow

Properties: define how to draw the data

Interactor: defines user interaction with the data

How to add a new data type

Extension for new data types:

➔ derive data class

➔ derive mapper

➔ create file I/O

➔ Register mapper / I/O handler at factory

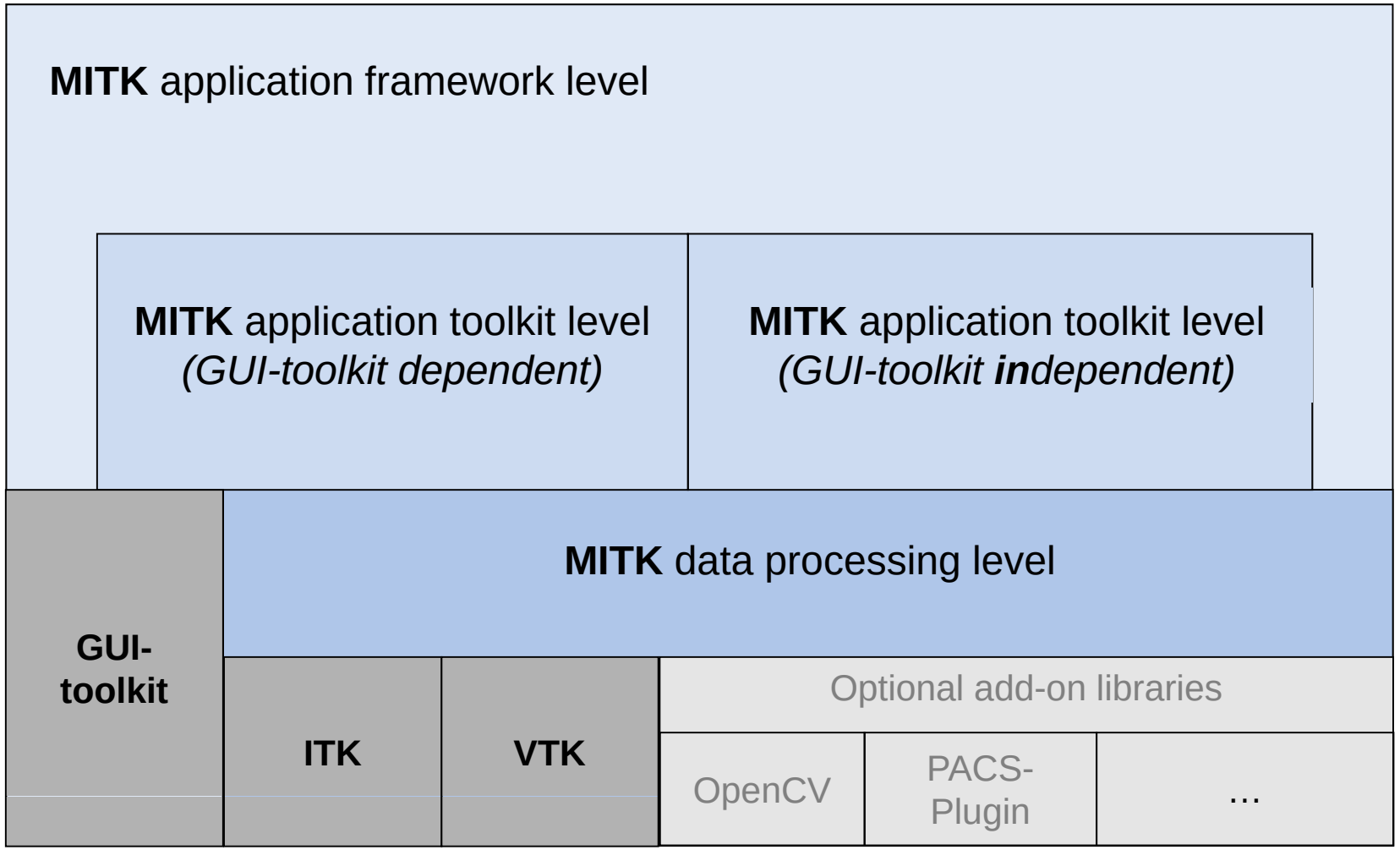
Example:

- attributed vessel graphs

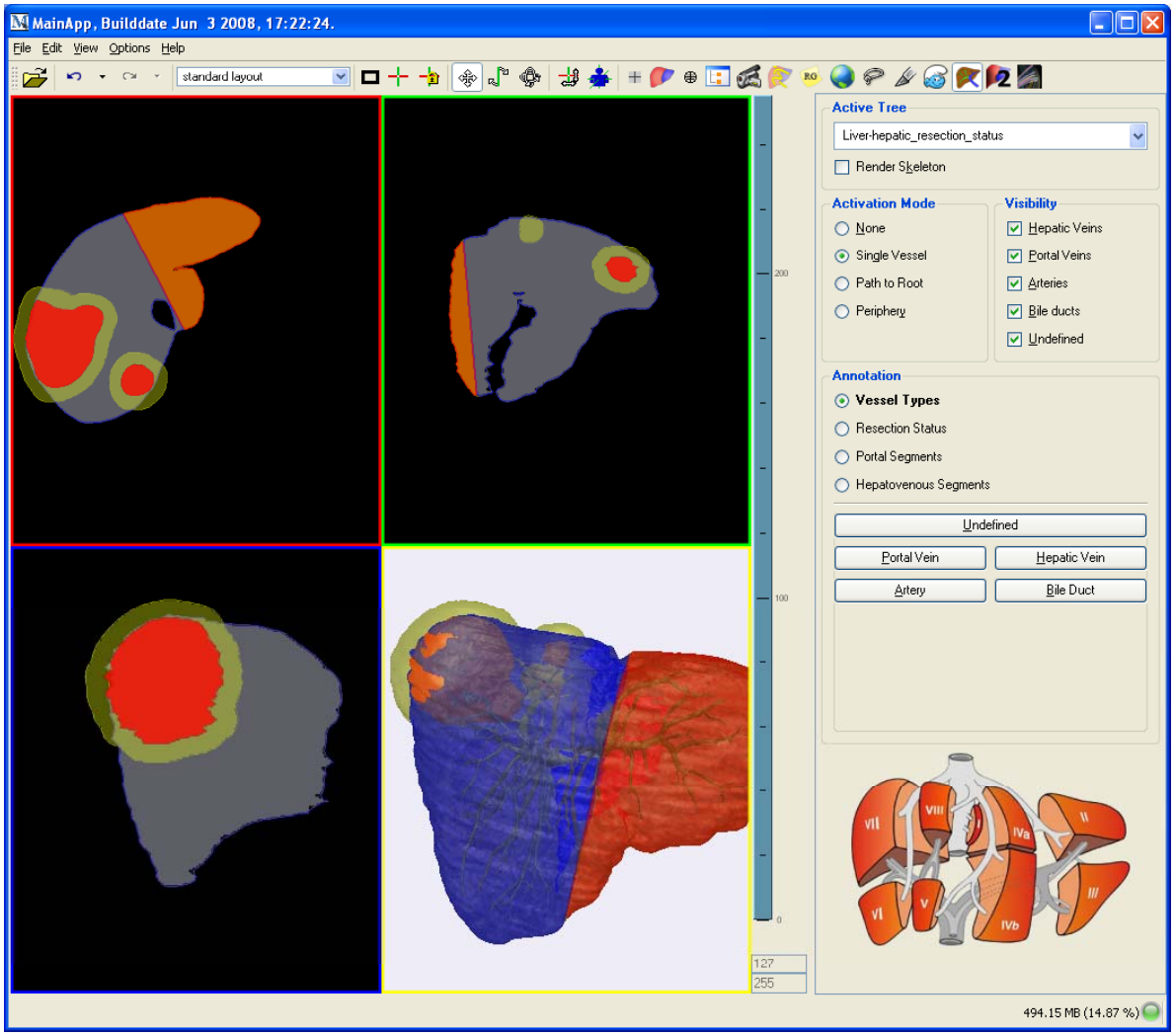
[DKFZ and University of Tübingen]

MITK Architecture

MITK Architecture



MITK - MainApp



MITK functionality modules

Functionality = a module with …

- an identification (icon/tooltip/…)
- a workspace area
- a control area
- a option dialog
- a help page (manual)
- the algorithmic implementation

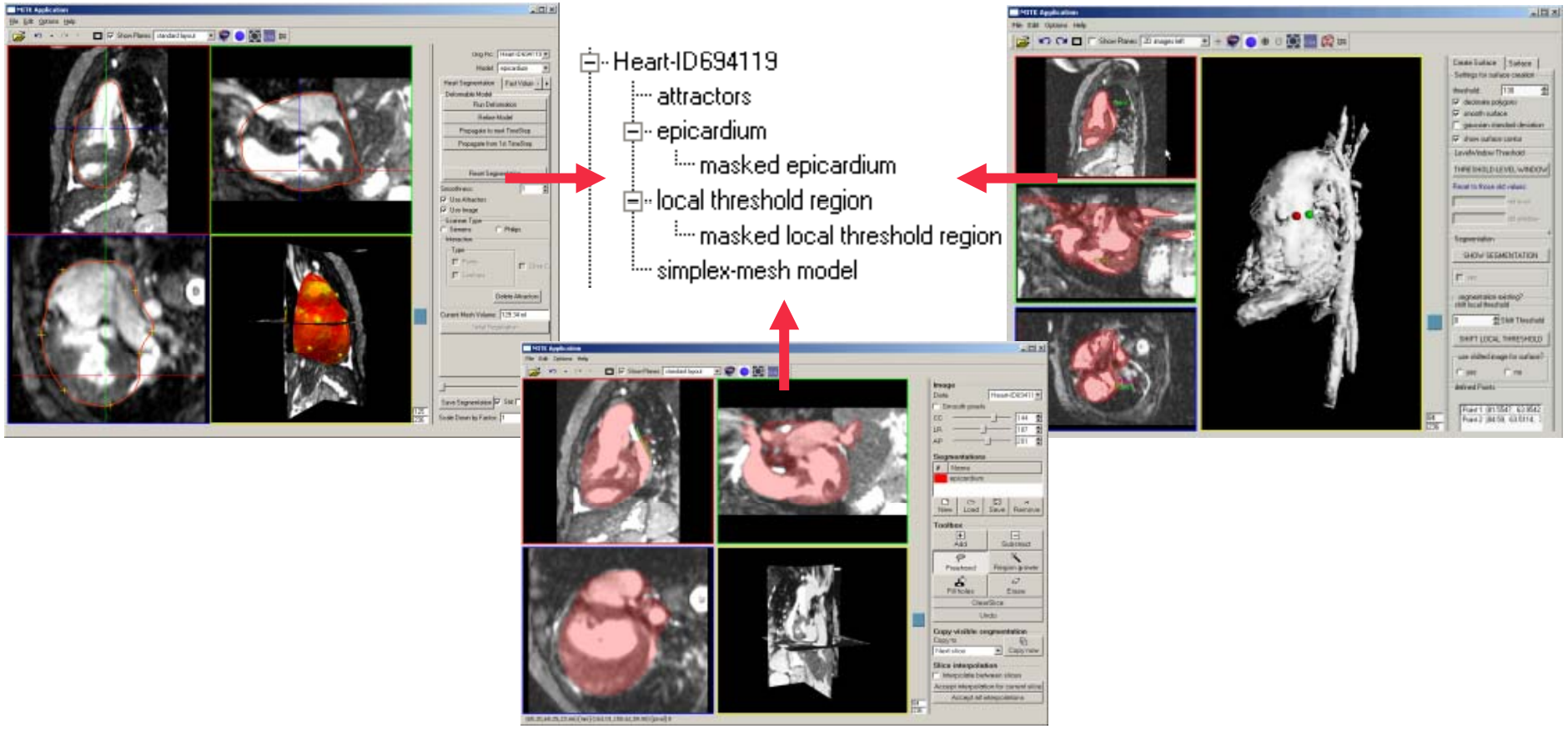
New platform:

control area = View

workspace area = Editor

Combining functionality blocks

- Functionalities are independent from each other
- They communicate via the data repository

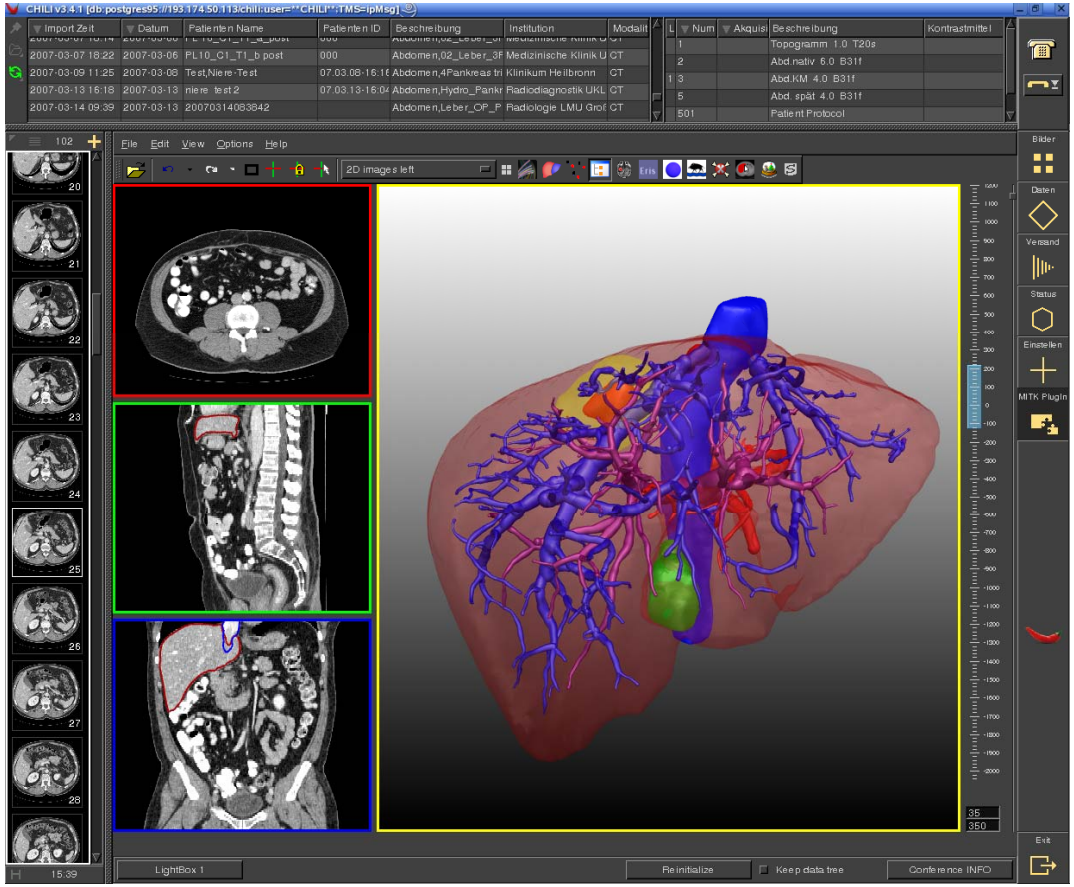


Embedding in PACS/tele-conferencing system

Integration in PACS/telemedicine system CHILI® as a PlugIn:

- PACS
- Connection to modalities
- DICOM import/export
- DICOM “unification”
- Data transfer
- Tele-radiology
- Management of results from image processing

➔ facilitates clinical integration



ITK, VTK, MITK:
Gemeinsamkeiten

Vorteile einer Standardisierung

- Kombination von neuen mit bereits existierenden Algorithmen und Verfahren
- Schnelle Vergleichsmöglichkeit unterschiedlicher Algorithmen und Verfahren für eine spezifische Problemstellung
- Schnellere Entwicklungszeiten und Verfügbarkeit der Ergebnisse
- Verbesserung der Softwarestabilität und -qualität

Gemeinsamkeiten

- Toolkits
- Objekt-orientierte Klassenbibliotheken
- C++
- Unterstützung vieler Compiler
- Plattform unabhängig
- GUI-Toolkit unabhängig
- Open source

Gemeinsamkeiten: Überblick

- Download und Build-Prozess
- Instantiierung und Pointer
- Daten-Pipeline
- Kommunikation
- Software-Prozess und Dokumentation

Download und Build

Voraussetzungen

C++ Compiler:

GCC 4.x

MinGW
VC++ 8 2005
VC++ 9 2008

Plattform-unabhängiges Make-Tool:

CMake

Projekt / Makefiles für

- die eingesetzte Plattform
- den verwendeten Compiler
- die Entwicklungsumgebung.

CMake

- Skriptsprache zur Beschreibung der
  Bestandteile eines (C++)-Projekts:
  Quelltexte, Header, Bibliotheken
- Getrennte Dateibäume für Quelltexte und Binärdateien
- Plattformen:
  Windows (Visual Studio, Borland, MinGW …), Linux,
  Mac OS X, seit CMake 2.6: Eclipse
- Erzeugt spezifische Projekt/Makefiles (nicht nur) für die
  Übersetzung der Programme und Bibliotheken
- Auch:
  Testgenerierung und -steuerung,
  Aufruf von Doxygen, LaTeX etc.
  Erstellung von Installern …

CMake Befehle

- **SET**(VAR [VALUE])
- **MESSAGE**(“Wert von VAR: ${VAR}”)
- Listen:
  LIST(APPEND VAR “NochEinWert”)
- Ausführbares Programm erstellen:
  ADD_EXECUTABLE(MeinProgramm Quelle1.cpp Quelle2.cpp)
- Bibliothek erstellen:
  ADD_LIBRARY(MeineLib LibQuelle1.cpp …)
- **TARGET_LINK_LIBRARIES**(MeinProgramm MeineLib)
- **INCLUDE_DIRECTORIES**(<Pfad für c++ header>)
- Einbindung von Bibliotheken, z.B.
  FIND_PACKAGE(Qt)

Weiter geht‘s mit einer kurzen Demo …

Zweiter Teil:
Grundkonzepte

Instantiierung von Klassen und Pointer

DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Instantiierung

Instantiierung von ITK/VTK/MITK-Klassen:

Statt

~~new className;~~

className::New();

Instantiierung

Grund für “className::New()” statt “new”:

- Factory für Instantiierung
- erlaubt dynamischen Austausch von Klassen

```
thisClass* thisClass::New()
{
  vtkObject* ret = vtkObjectFactory::CreateInstance(“thisClass”);
  if(ret != NULL)
      return static_cast<thisClass*>(ret);
  else
       return new thisClass;
}
```

[bei ITK/MITK ähnlich]

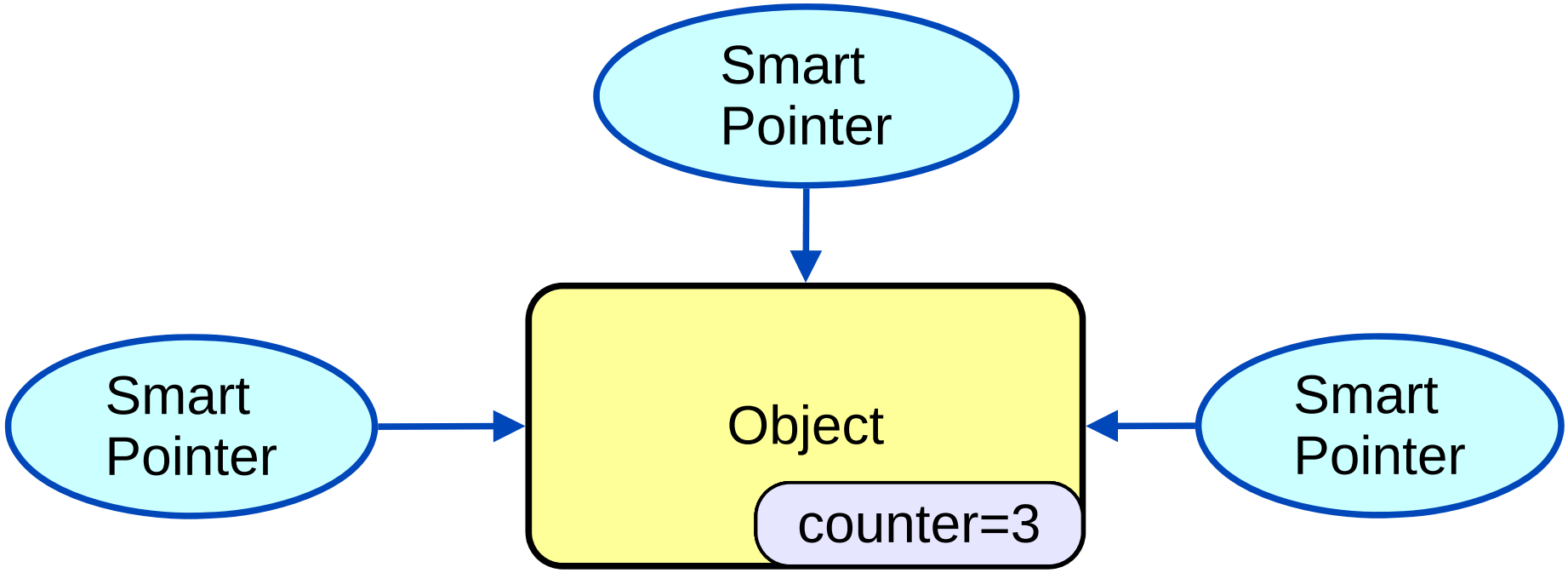
Instantiierung

```
//VTK:
vtkRenderer *renderer =
  vtkRenderer::New();
//ITK/MITK:
itk::Command::Pointer command =
  itk::Command::New();
```

ACHTUNG:

“*” statt “::Pointer” *bei Instantiierung* führt sehr bald zum Absturz!!

::Pointer ist ein Smart Pointer



Self - Delete

VTK: “Manuelles” Reference Counting

```
//VTK:
vtkRenderer * renderer =
  vtkRenderer::New();              //count==1

vtkRenderer * ref_to_renderer;
ref_to_renderer = renderer;        //weiterhin: count==1
ref_to_renderer->Register(...);    //count==2

renderer->Delete();                //count==1, (noch) kein
                                   //wirkliches delete!
ref_to_renderer->Delete();         //count==0, delete!
```

Daten-Pipeline

Daten-Pipeline

Objekt-orientierter Ansatz:

- Daten-Klassen

  und

- Algorithmen-Klassen

➔ Algorithmen nicht als Funktionen, sondern als Klassen realisiert !

Daten-Pipeline

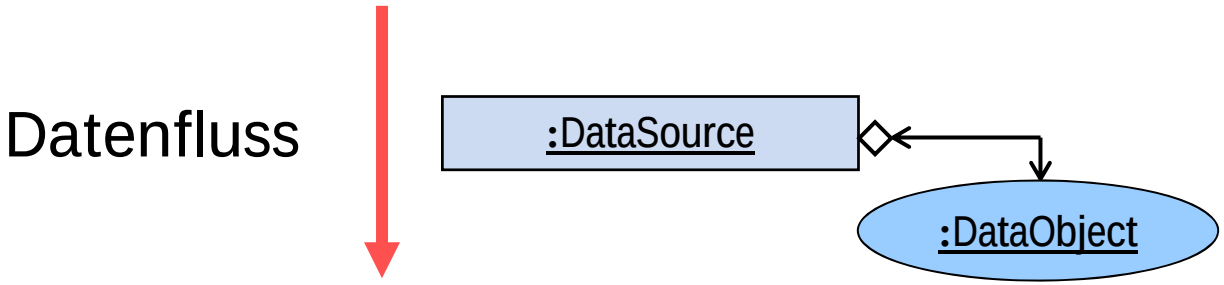
Algorithmen-Klassen:

- Oberbegriff: **ProcessObject**
- Daten-Erzeuger: **Source**, z.B. File Reader
- Daten-Verarbeiter: **Filter**
  (sind somit ebenfalls Sourcen)

Daten-Pipeline

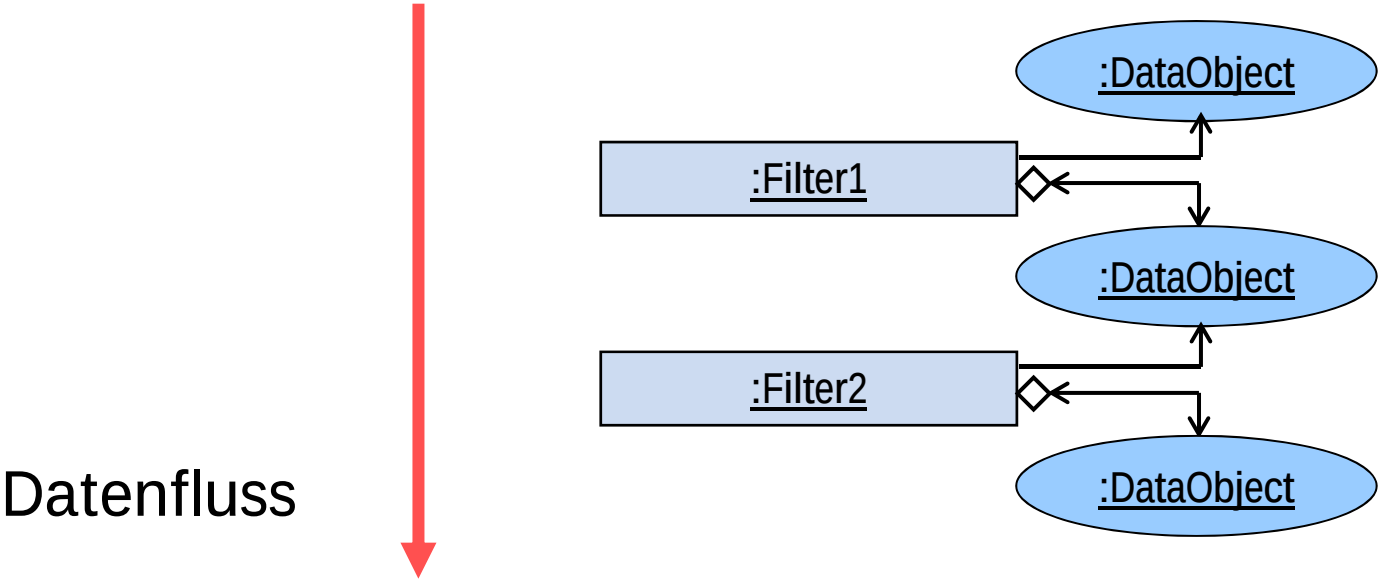
Sourcen **besitzen** ihre Ausgabe-Daten-Objekte!

Umgekehrt **kennt** das Ausgabe-Daten-Objekt **seinen Erzeuger** (Source)!



Daten-Pipeline

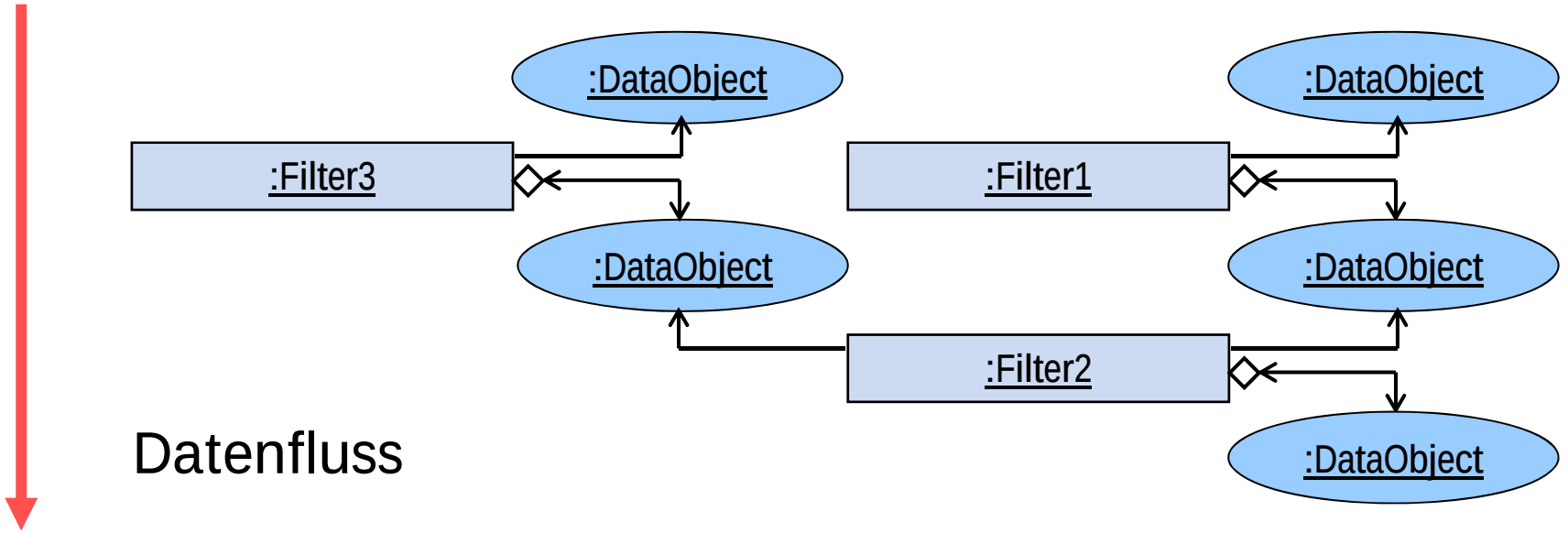
Filter **kennen** zudem ihre Eingabe-Daten-Objekte:



➔ **Daten-Pipeline!!**

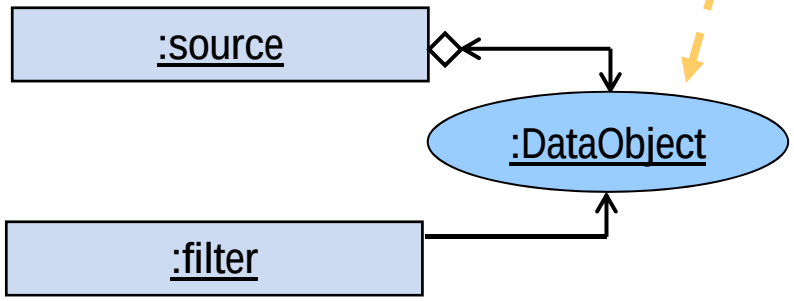
Daten-Pipeline

Filter können auch **mehrere** Eingabe-/ Ausgabe-Daten-Objekte haben:



Verbinden von Filtern

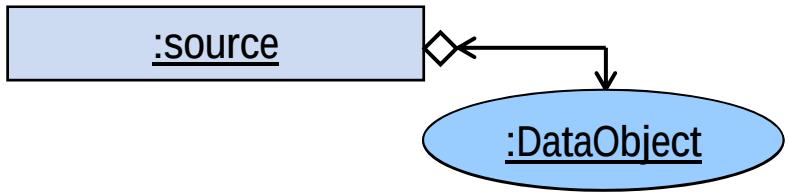
```
SourceType::Pointer source = SourceType::New();
FilterType::Pointer filter = FilterType::New();
filter->SetInput(source->GetOutput());
```



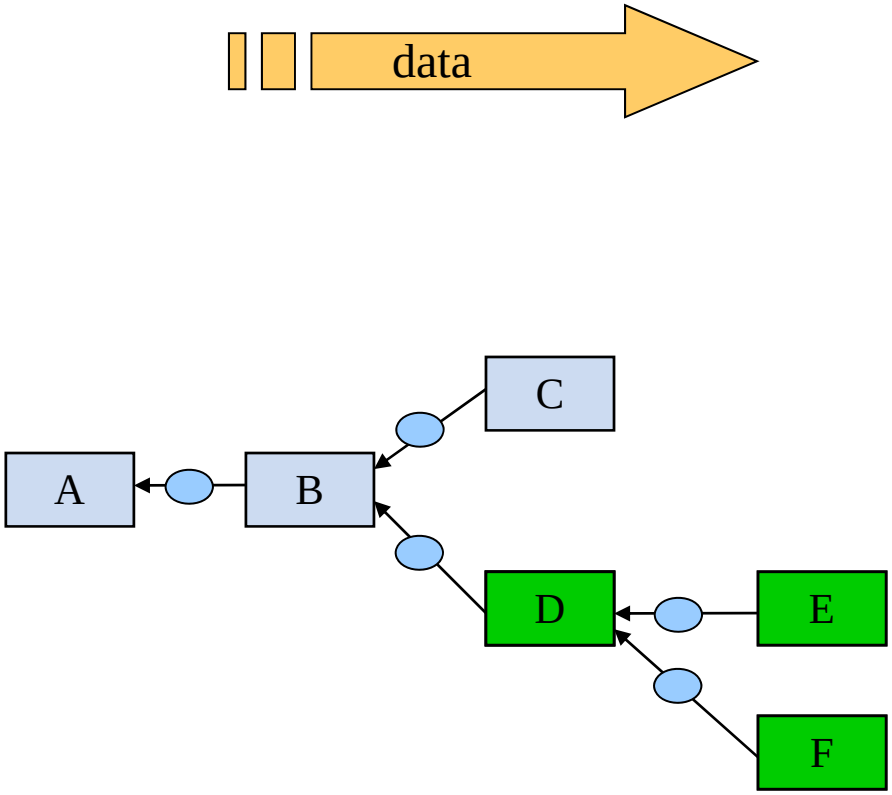
➔ ***Daten-Objekte meist “unsichtbar”!***

Update/Leere Daten-Objekte

```
SourceType::Pointer source = SourceType::New();
// Achtung: Daten-Objekt ist zunächst leer!!
// Garantie für aktuelles Daten-Objekt: Update()!
source->Update();
```



Pipeline update …

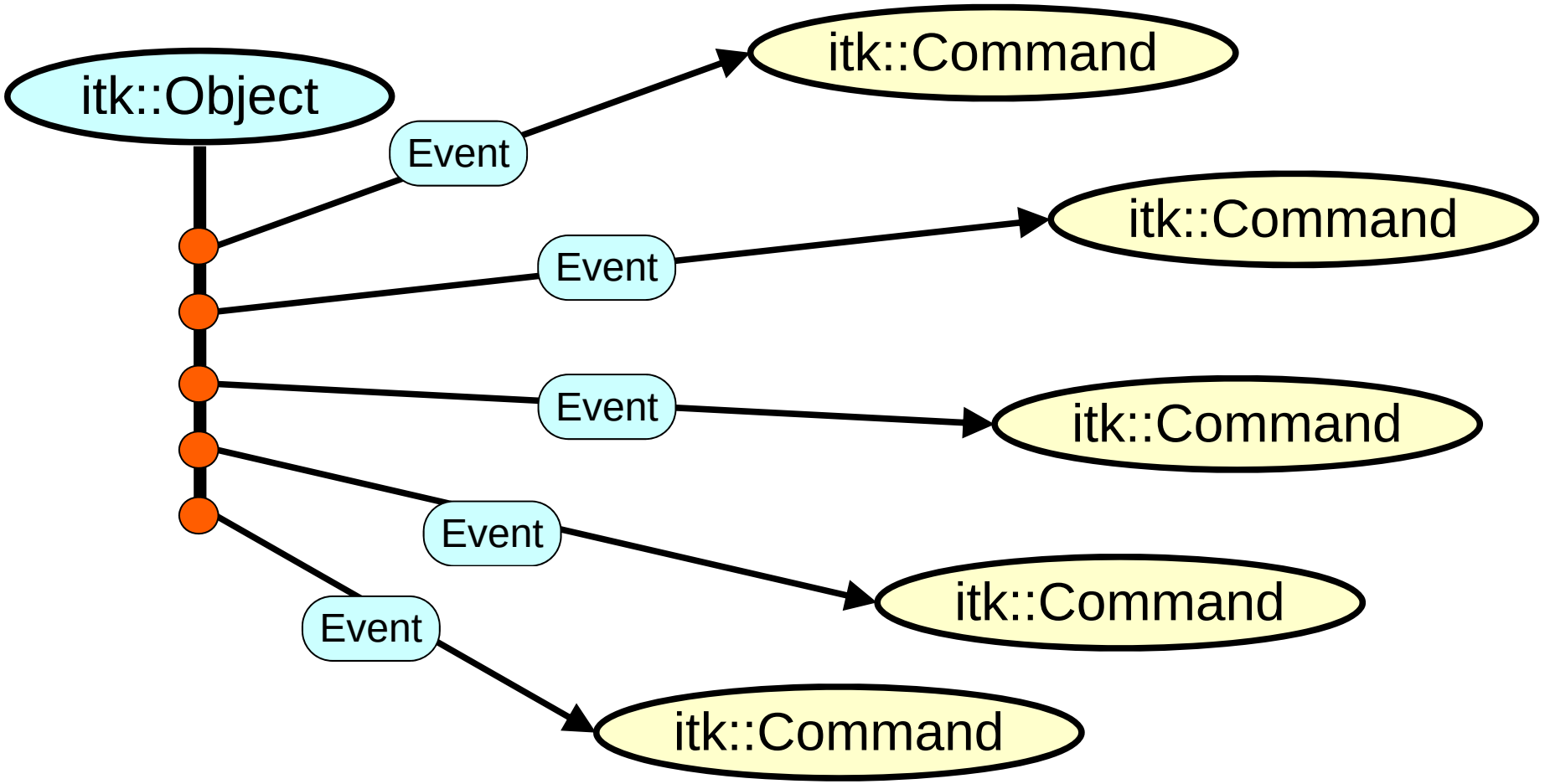


➔ Keine unnötigen Neuberechnungen !!

Kommunikation:
Events, Observer, Commands

dkfz.
DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Kommunikation (hier ITK/MITK, ähnlich in VTK)

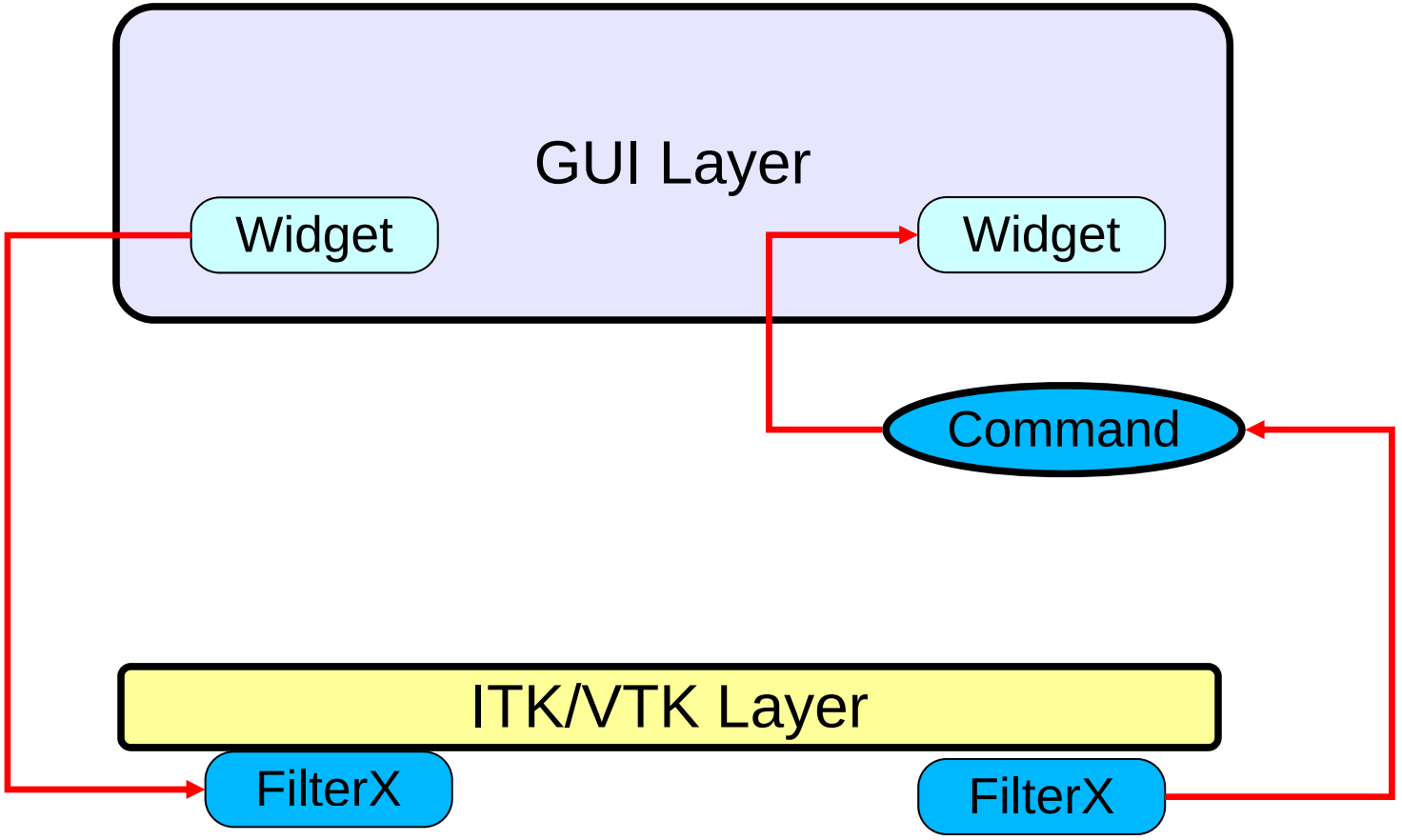


Kommunikation

```
// Anmeldung eines “commands” (Empfänger)
object->AddObserver ( event, command );

// Verschicken eines Events an die angemeldeten
// Empfänger
object->InvokeEvent( event );
```

Kommunikation GUI ⇔ ITK/VTK



Adaptor Classes: Connecting ITK, VTK data classes to MITK

dkfz. DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

Access MITK images as ITK images

- ITK images are templated
- mitk::Image is not templated

```
// access method
template < … >
MyAccessMethod( itk::Image<…>* itkImage, … )
{
…
}

// calling the access method
AccessByItk(mitkImage, MyAccessMethod, …)
```

Example code in mitk/Applications/Tutorial/Step6.cpp

Converting images ITK⇔MITK

- mitk::CastToItkImage(mitkImage, itk::Image<…>)
  - converts data type if neccessary
  - otherwise references memory of data array
  - dimension of itk::Image must equal dimension of mitkImage
- mitk::Image::Pointer
  mitk::ImportItkImage (itk::Image<…>,*update*=true)
  - references memory of itk::Image

Connecting MITK images to VTK

… not a conversion, just accessing:

- **vtkImageData***

  mitk::**Image**::**GetVtkImageData**(int time=0)

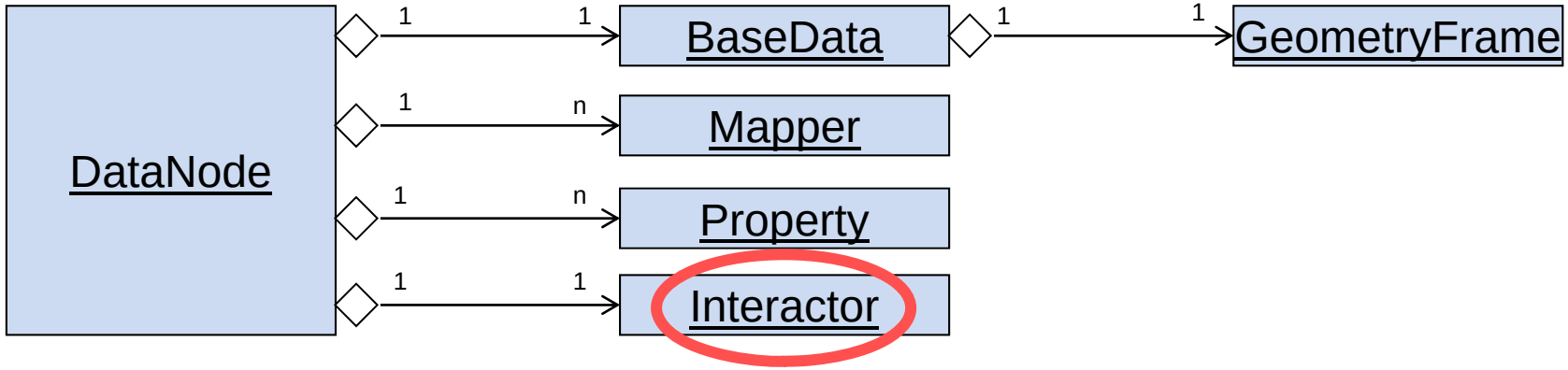
Surfaces MITK ⇔ VTK

Again: not a conversion, just accessing:

- **vtkPolyData***
  mitk::**Surface**::**GetVtkPolyData**(int time=0)
- mitk::**Surface**::
  SetVtkPolyData(**vtkPolyData***, int time=0)

Interaction

Interactors



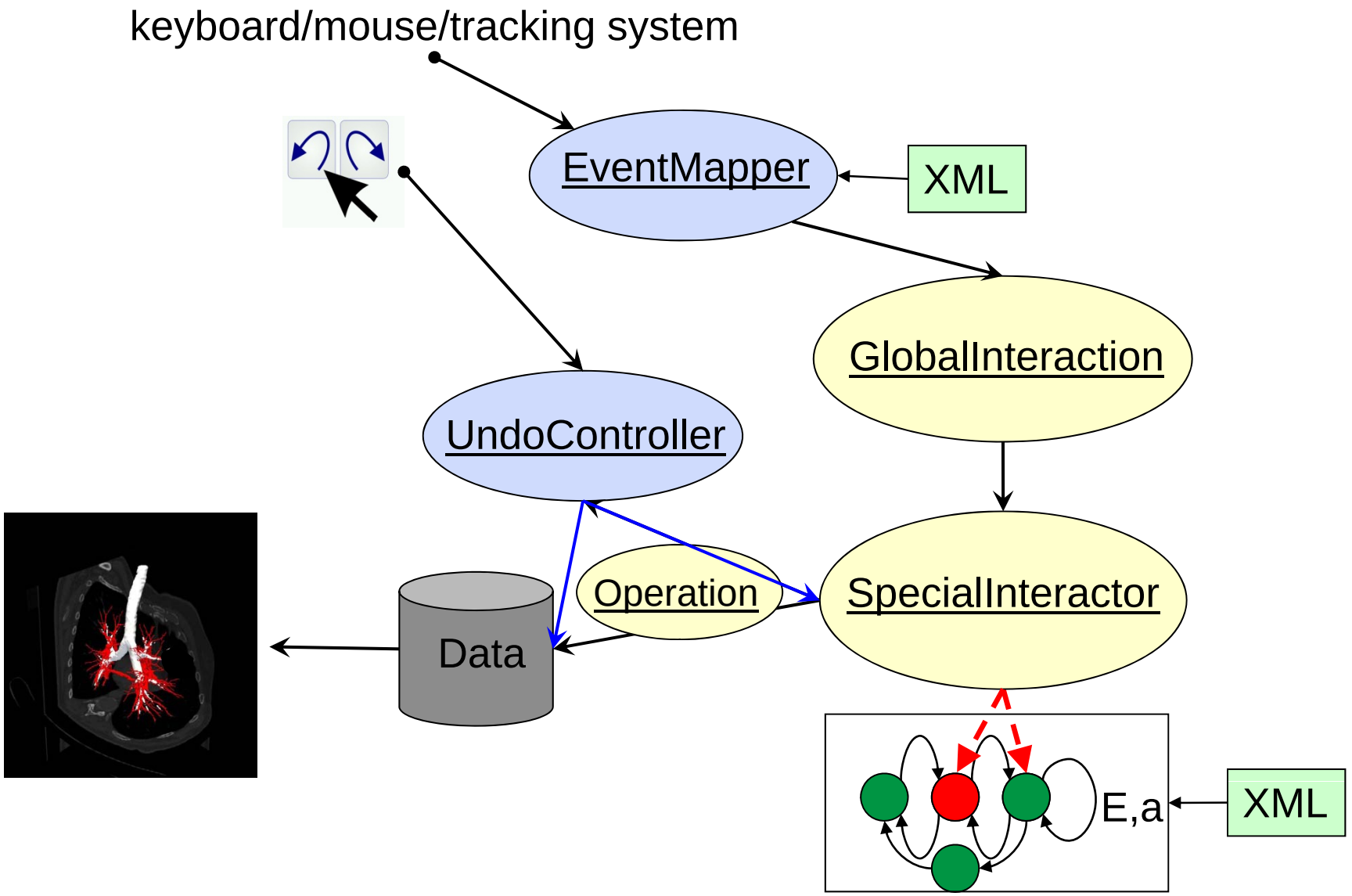
Interactors:

➔ behavior defined in state-machines

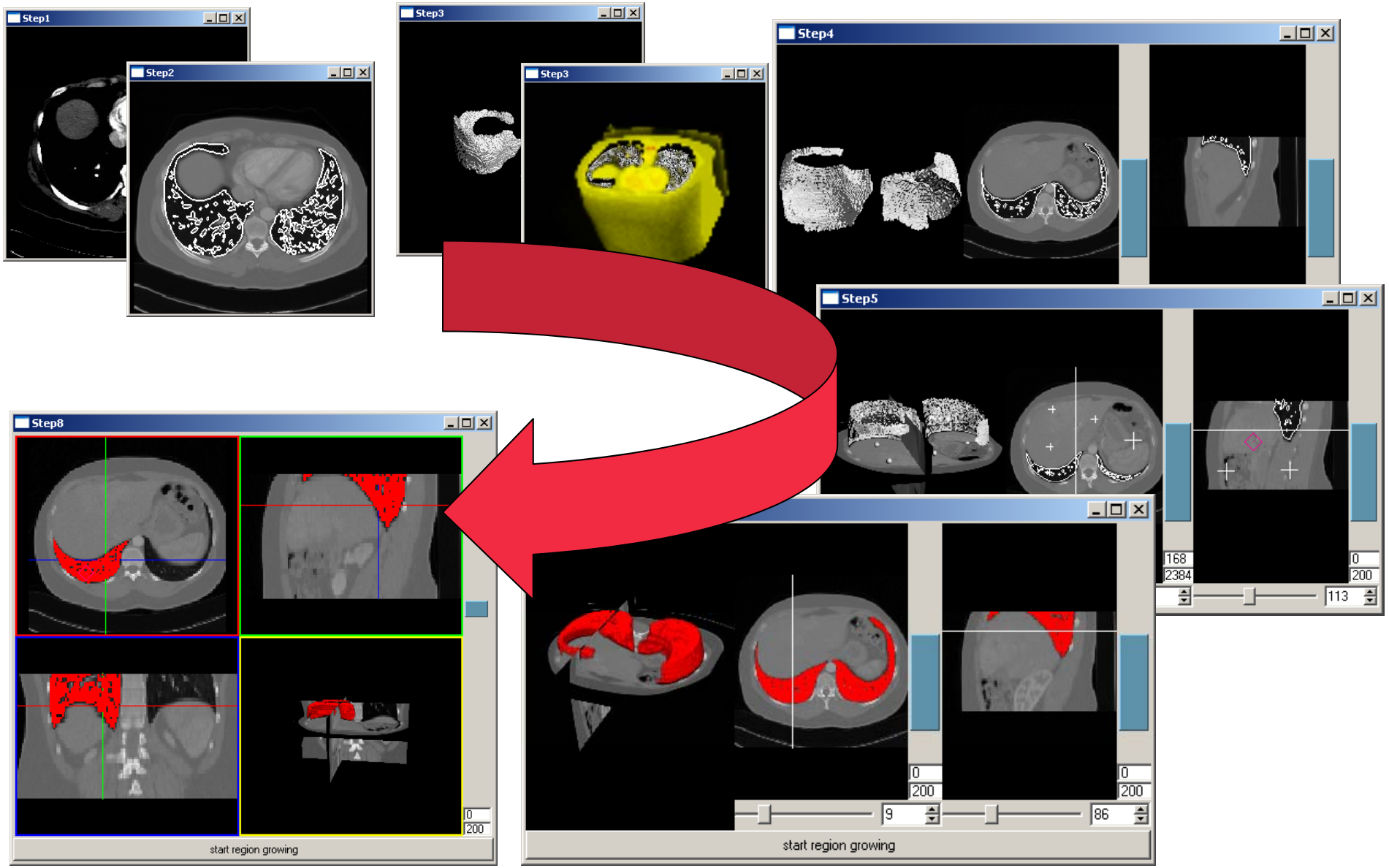
➔ undo-/redo concept

➔ dimension/geometry-independent definition:
(often) identical interaction code for 2D and 3D

Interaction Scheme



9-step tutorial



Vielen Dank!

Fragen?

Kaffeepause …

Ausblick

Today's Open-Source Toolkits

Kitware (ITK,VTK)

NAMIC (Slicer)

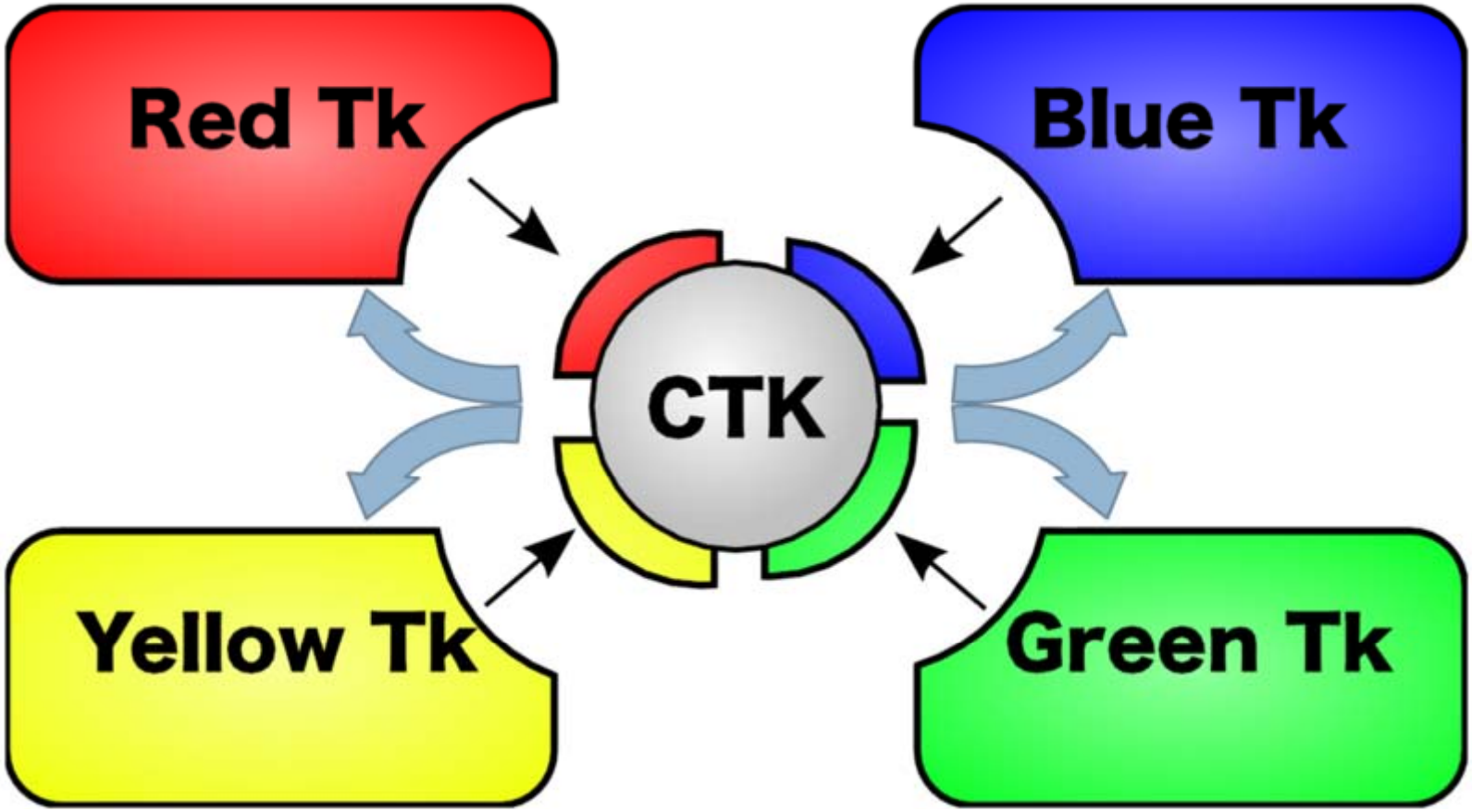
INRIA (MedINRIA)

DKFZ (MITK)

OpenMAF.org (MAF 3)

Siemens (OpenXIP)

The CTK Concept



Starting points: DICOM I/O

- CTK DICOM modules based on DCMTK
- Support for more data types (surfaces, segmentations …)
- DICOM communication methods
- WG 23: standard for application hosting

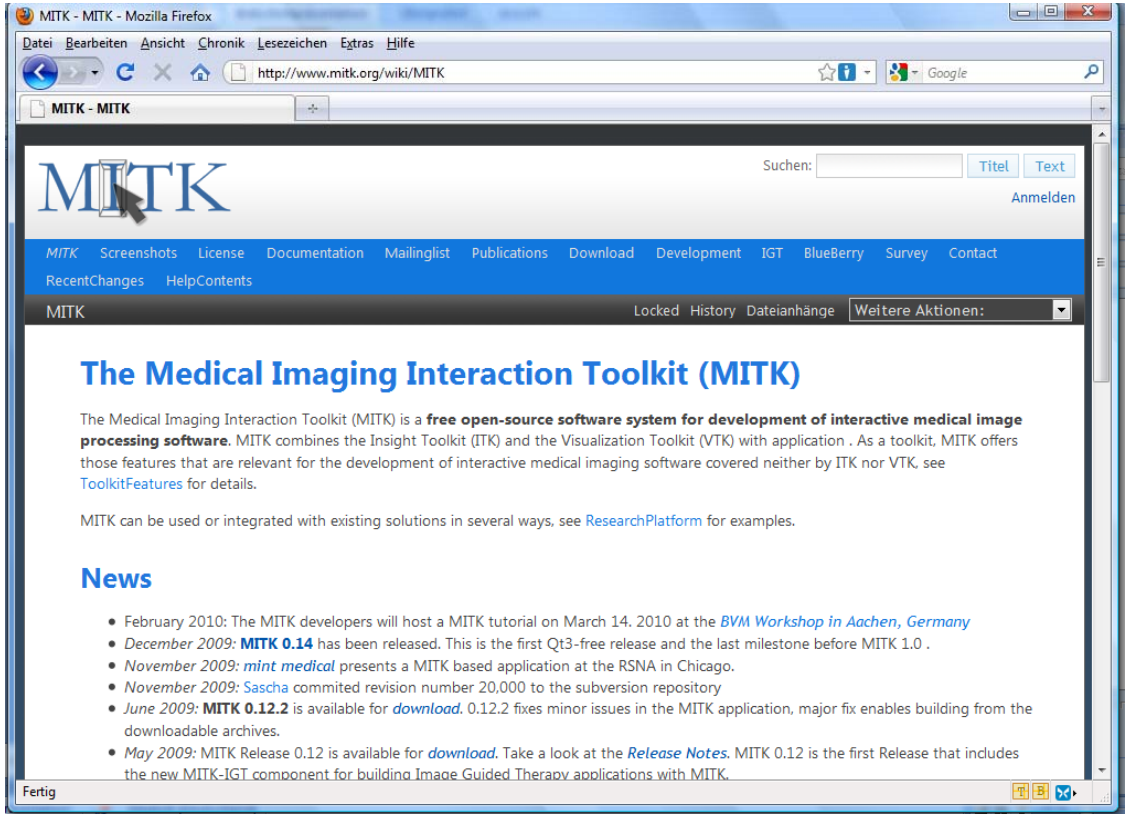
Common Toolkit

- Erstes Treffen Juni 2009 in Heidelberg
- Drei weitere Workshops in Oxford, Chicago und San Diego
- Erstes „Hackfest“ vergangene Woche in Heidelberg
- Weitere Infos:
  http://commontk.org
  http://github.com/pieper/CTK

How to get started

Download options:

- anonymous svn:
  http://svn.mitk.org/trunk/mitk/
- zipped archive
  https://sourceforge.net/projects/mitk/
- Windows Installer
- Demo Application
  http://3m3.mitk.org



Dokumentation & Tutorial:

http://docs.mitk.org/

Support:

- mitk-users Mailingliste
- http://bugs.mitk.org
- Kommerziell

mint medical

Das wars …

Vielen Dank!

Fragen?

Daniel Maleike d.maleike@dkfz.de
Michael Müller michael.mueller@dkfz.de
Marco Nolden m.nolden@dkfz.de
Alexander Seitel a.seitel@dkfz.de