Introduction to the GenICam Standard

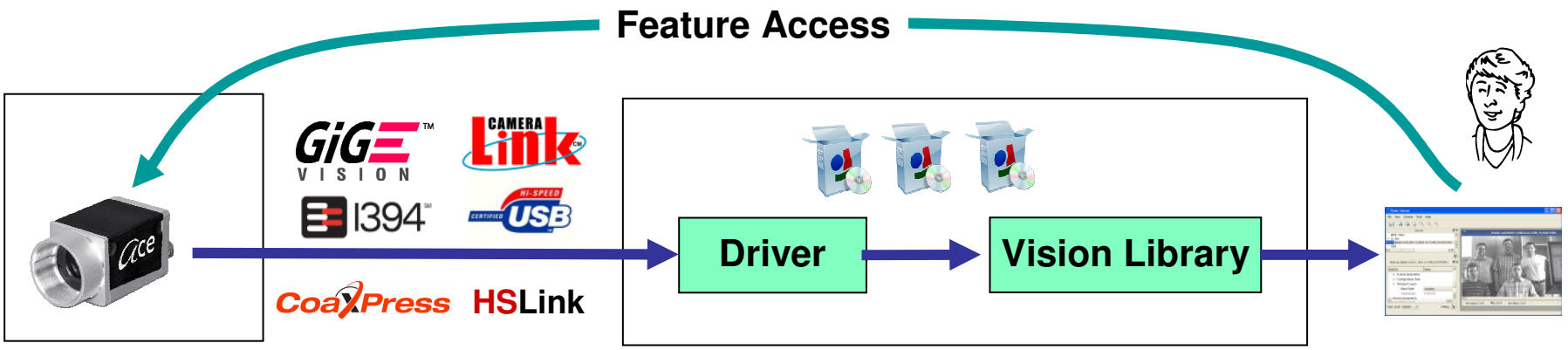
Dr. Fritz Dierks
Chief Engineer & Head of SW Development
Basler AG

Chairman of the GenICam standard committee

Why GenICam?

GEN<i>CAM

provides **plug&play** to machine vision cameras



Camera Vendors

PC Software Vendors

Customers

emva
european machine vision association

GenICam Members

(2006) : 9 → 20 → 47 → 60 →77 : (Jan 2010)

Investments in GenICam

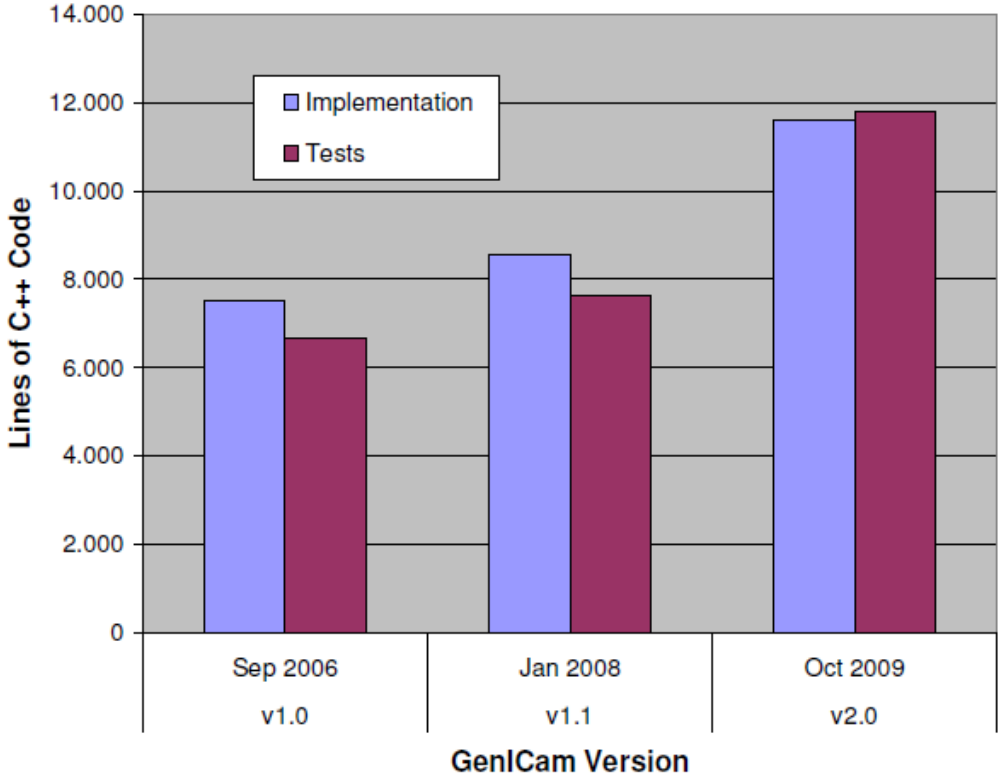
Committee Work

- 7 years of intense work
- 17 international meetings
- ~15 companies per meeting*)

Common Code Base

- Used by nearly all companies but not part of the standard
- Written in C++
- Supports Win32 / Win64 with Visual Studio 7.1 / 8.0 / 9.0
- Supports Linux32 / Linux64 with gcc>=4.0, glibc>=2.3.5
- Strict focus on quality

*) since 2005

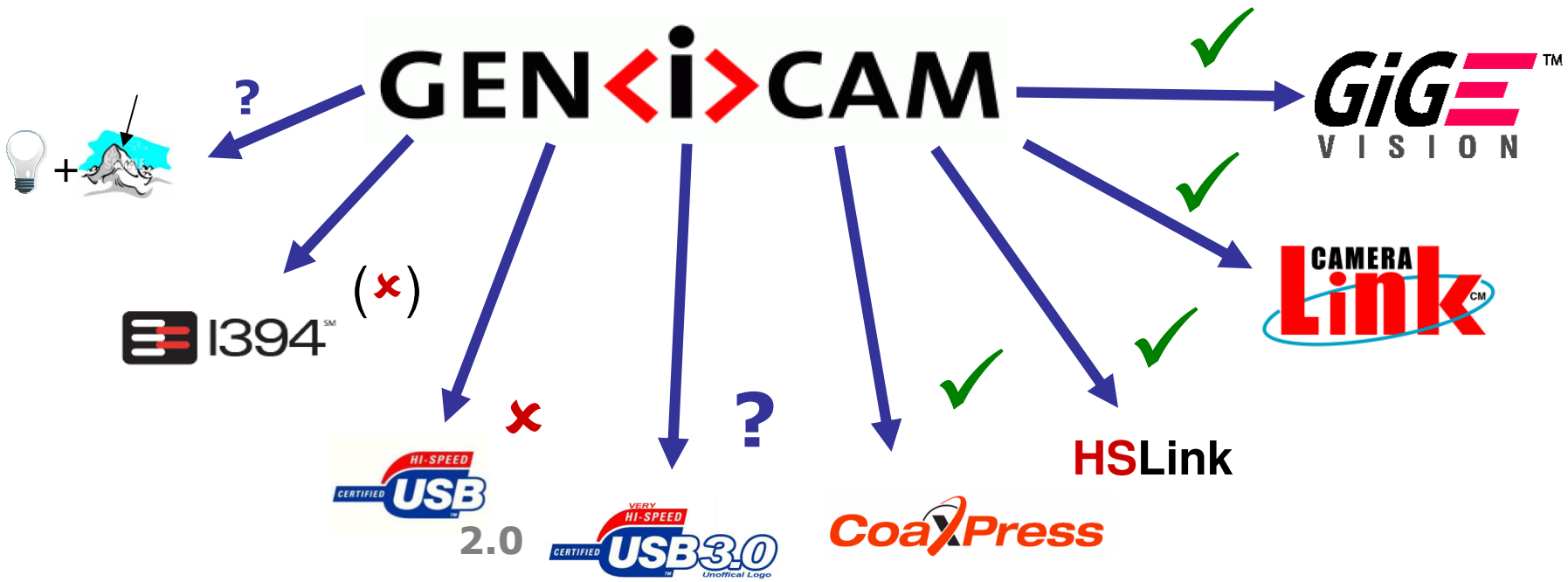


Investments**)

- Meetings >300 k€
- Common code base >500 k€

**) rough estimate; does not include product development

Interfaces Supporting GenICam



Cost for adding GenICam support*)

- Introducing GenICam for the first interface : ~50 k€
- adding another interface to existing GenICam support : ~ 10 k€

*) very(!) rough estimate

Some Questions

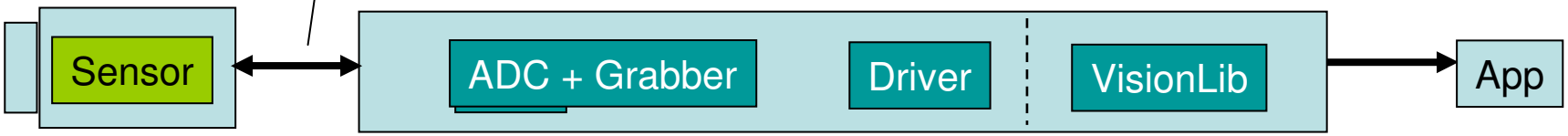
- **What makes GenICam attractive?**
  - Serves a market need
  - Has hit a window of opportunity
  - Has mechanisms to evolve quickly
- **Which Modules does GenICam Consist of?**
  - Camera Configuration (modules GenApi & SFNC)
  - Image Acquisition (module GenTL)
- **What is the Status and Roadmap for GenICam?**

➔ Let‘s have a look at the details

History of Camera Configuration

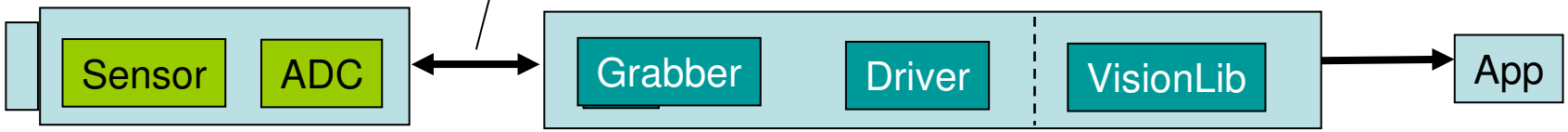
Standard = video signal + DIP switches
➔ **plug&play**, but limited video formats only

PAL NTSC



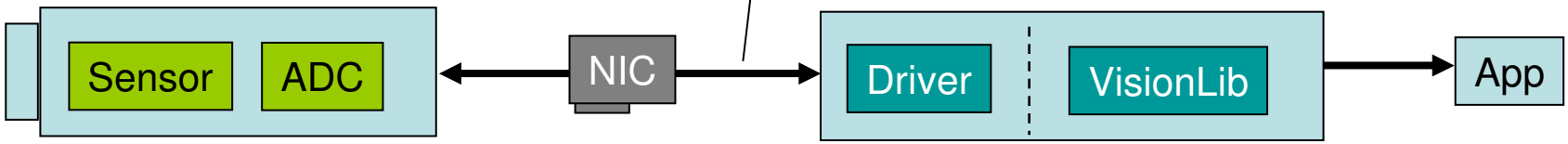
Standard = serial port, no defined protocol
➔ **no plug&play**

CAMERA Link



Standard = fixed register layout
➔ **plug&play**

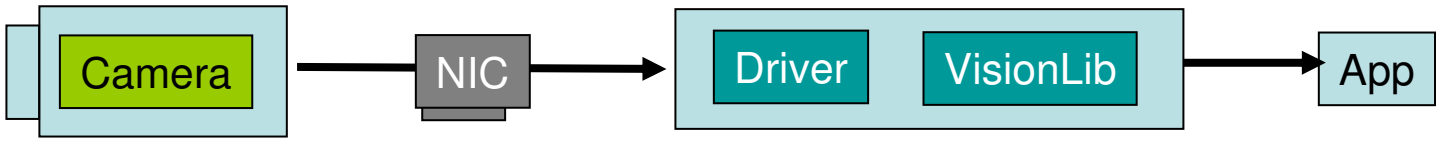
1394



Problems with Fixed Register Layouts (1/2)

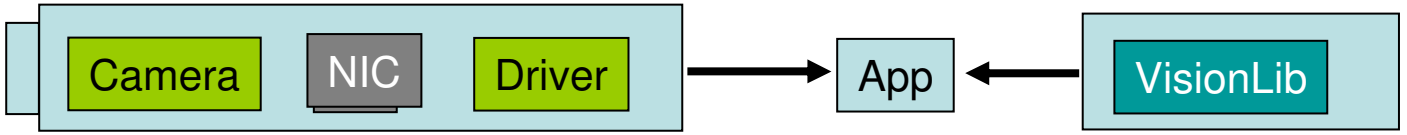
➔ **Missing Custom Feature Support**

No Business Model for Custom Feature Support



- Custom features require **expensive manual coding** in the driver
- It hardly makes sense for a driver vendor to support camera custom features. Example:
  Camera vendor : 400 cam/yr * 1000 €/cam = **400 k€**/yr ➔ sweet deal ☺
  Driver vendor : 400 license/yr * 100 €/cam = **40 k€**/yr ➔ sour deal ☹

Workaround: Cameras Come with their Own (Free) Driver



- Only for network based cameras
- Proprietary solution, no integration into vision library
- Free drivers puts a lot of pressure on driver/VisionLib business model

Problems of Fixed Register Layouts (2/2)

➔ Standard Defines Too Many Details

Fixed Register Layout Contains Lots of Implementation Details

(bit depth, feature inquiry, min/max/inc, ...)

➔ Slow Standard Evolution

– Exhaustive discussion about bits & bytes
– Each company is fighting for their specific layout
– Only really large companies can start a standard layout (1394 IIDC = Sony)

➔ No Migration Path from Custom to Standard Features

– New features are implemented as custom features for sake of speed
– If feature is later standardized and gets a different register layout
  → no adoption possible because of backward compatibility
– Proposing standard features makes not too much sense for a company

The Window of Opportunity

GigE Vision Standard

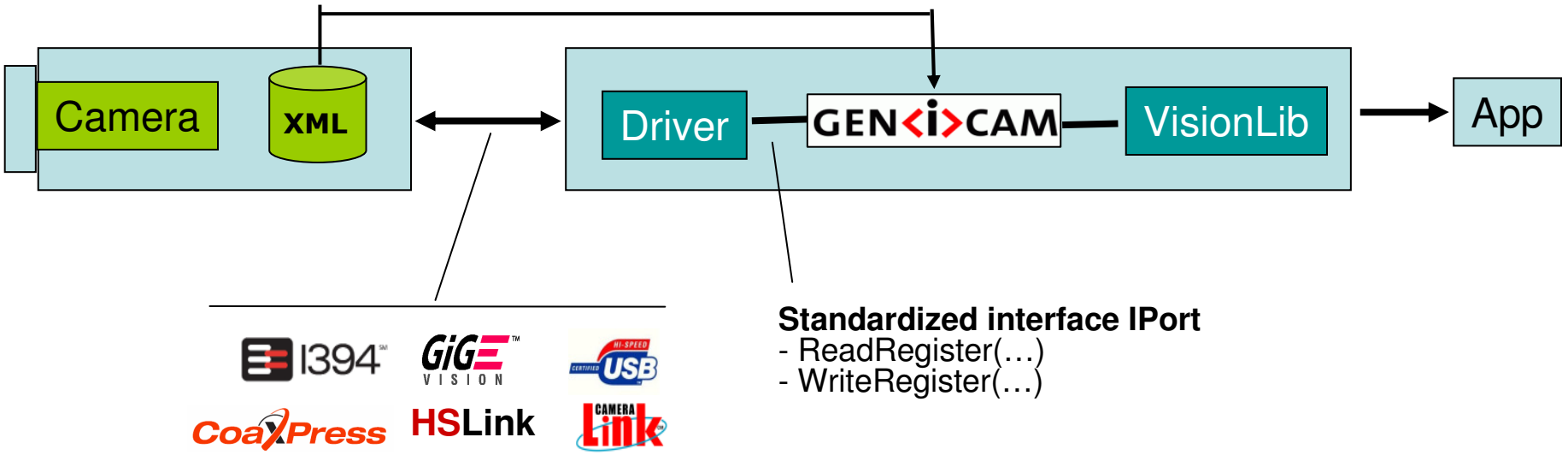
- Kick-off meeting June 2003
- Every company tried to get their proprietary register layout standardized
- After one year no conclusion was reached

➔ **committee was stuck ☹**

Escape Route

- Let every camera have their own register layout
- Define standard **features abstractly**
- Have a **camera description file** in **XML** format with describes how to map the abstract features to the registers

➔ **Birth of GenICam**



Standardized interface IPort
- ReadRegister(…)
- WriteRegister(…)

GenICam Modules GenApi and SFNC

GenApi Module

- Defines the XML language of the **camera description file**
- Supported **types**: Integer, Float, Enumeration, Bool, String
- Each type corresponds to an **interface** with methods like GetValue, SetValue, GetMin, GetMax, etc.
- Camera has a set of **features**
- Each feature has a name, a type and a meaning ➔ **abstract**
- Description syntax is the same for **custom** and **standard** features

➔ **Full Custom Feature Support**

Example

- **Name** = „Gain“
- **Type** = IInteger
- **Meaning** = camera amplification

SFNC*) Module

- Defines a set of abstract features forming the ideal camera
- No details, just the name, type and meaning
- List has grown to >400 features

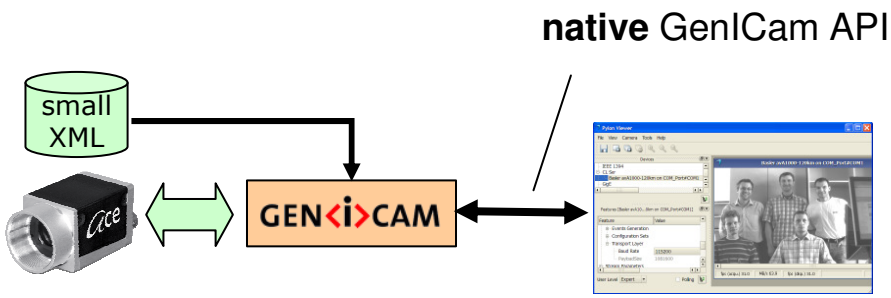
➔ committee was un-stuck ☺

*) SFNC = Standard Feature Naming Convention

How Things Worked Out

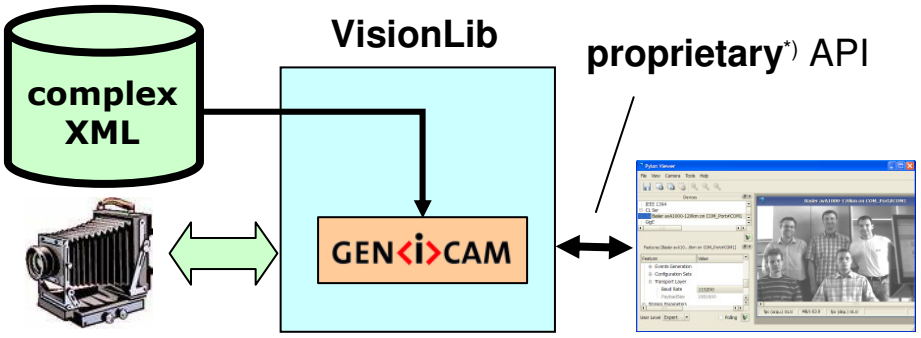
Original Assumption

- Customers use the native GenICam API
- XML file contains a ~1:1 mapping of registers to features



What Happened in Reality

- Library vendors used GenICam as **engine under the hood**
- Customers got the functionality of GenICam but through the libraries' native API
- XML file is used to map legacy registers to SFNC features



*) some use GenICam natively; many have a back-door

How GenICam can Evolve Very Fast

Voting Rules

- Membership to GenICam committee is free
- 1..2 meetings per year; homework between meetings
- Only companies contributing homework can vote*)
  ➔ Who invests money gets in the driver seat

Migration Path from Custom to Standard Features

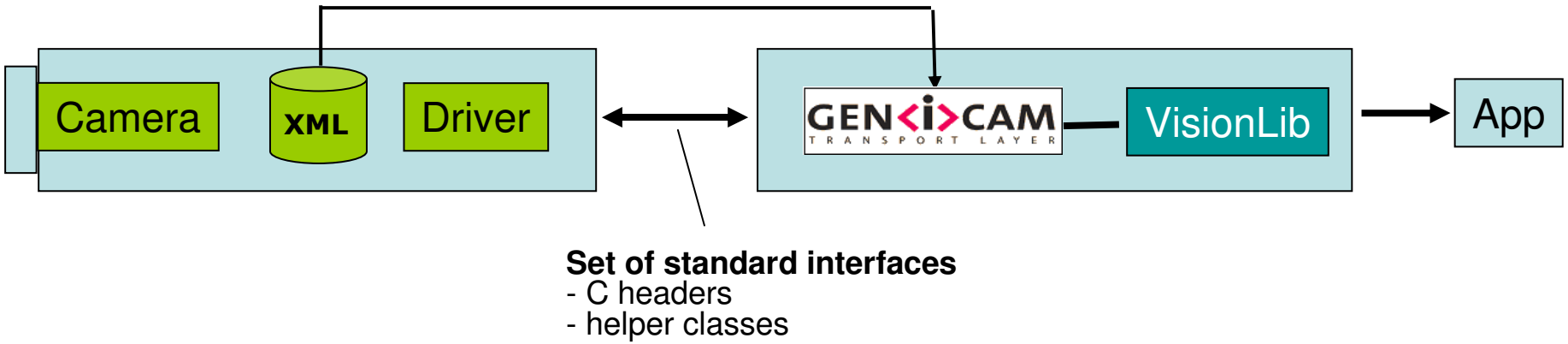
- New features are implemented by some company as custom feature
  ➔ **immediate business**
- The feature can be added to SFNC list later ➔ **adds proven features**
- Custom features become standard by changing an attribute in XML file

*) GigE Vision and CameraLink borrowed these rules recently

GenTL Module – The Grab Interface

Modules:

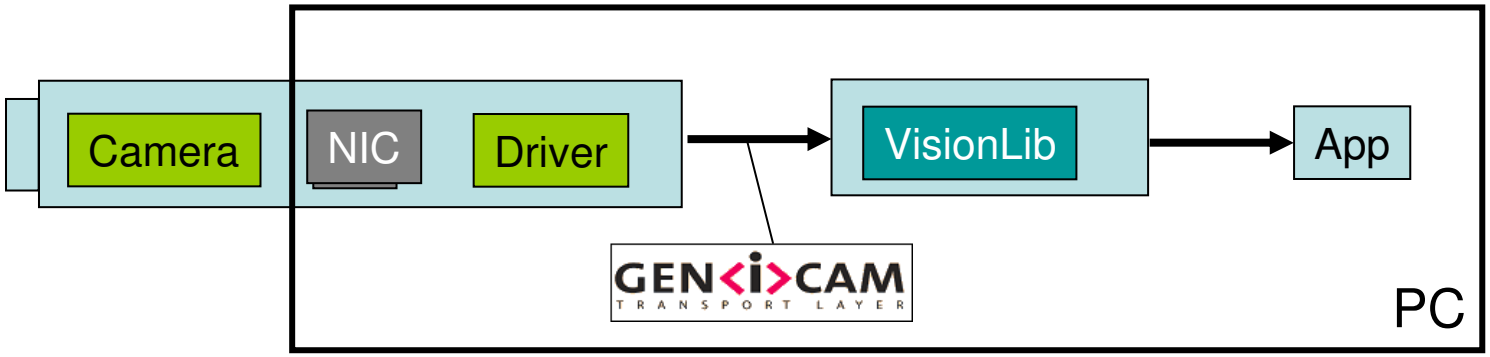
- **GenApi/SFNC** : camera configuration
- **GenTL** : enumerating devices, retrieving XML file, grabbing images



Why GenTL?

- Typically camera vendors have drivers for their own products
- Integrating a driver into an image processing library requires quite some effort
- With GenTL comes plug&play: just install the driver and the library can use it

The GenTL Business Case



Why is there so little GenTL Support?

- Splits responsibility on the PC side (support)
- Operating system support depends on camera vendor
- Once most library vendors have their own driver there is not much GenTL demand any more
- GenTL missed the first window of opportunity (by Nov 2008 everybody had a driver)

Now there is a New Window of Opportunity!

- Lots of new interfaces are evolving (CoaXPress, CameraLink HS, USB 2.0/3.0, LightPeak, …)
- It is too expensive for everyone to develop their own drivers / frame grabbers
- Solution: Make basic GenTL support mandatory to the transport layer standards
- Benefit: Immediate access to image processing libraries even if the user base is still small
- Good news: there is growing activity!

➔ Overcome the Chicken & Egg problem

Status and Roadmap

GenICam v2.0

- Released November ‘09
- Maintenance release v2.0.1 February ‘10
- Contains GenApi v2.0, SFNC v1.3, GenTL v1.1

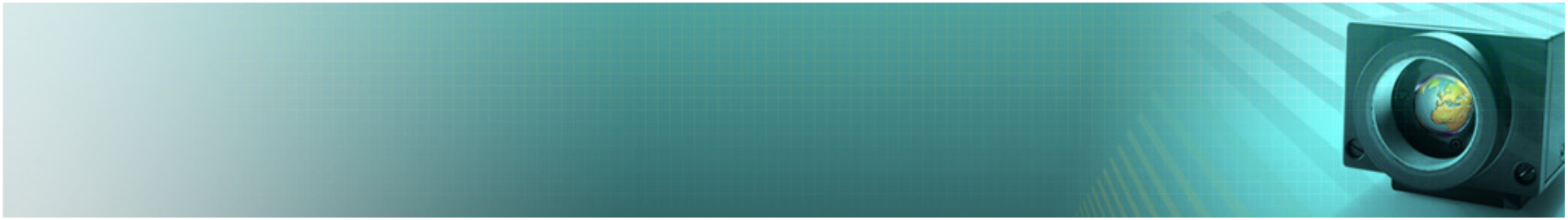
GenICam v2.1 Release Candidate

- GenApi → maintenance
- CLProcotol v1.0 → CameraLink support
- SFNC v1.4 → Updated
- GenTL v1.2 → Updated

New

What comes next?

- Improving documentation (extending tutorial)
- Supporting more compilers (VS100)
- Supporting more platforms
- Improving support for frame grabber based system ➔ any feedback welcome ☺



Dr. Fritz Dierks
Chief Engineer
& Head of SW Development

Basler AG

An der Strusbek 60-62
22926 Ahrensburg
Germany

Phone: +49-4102-463-381
Email: friedrich.dierks@baslerweb.com

www.baslerweb.com