Karlsruhe Institute of Technology

A UFO Framework Prototype

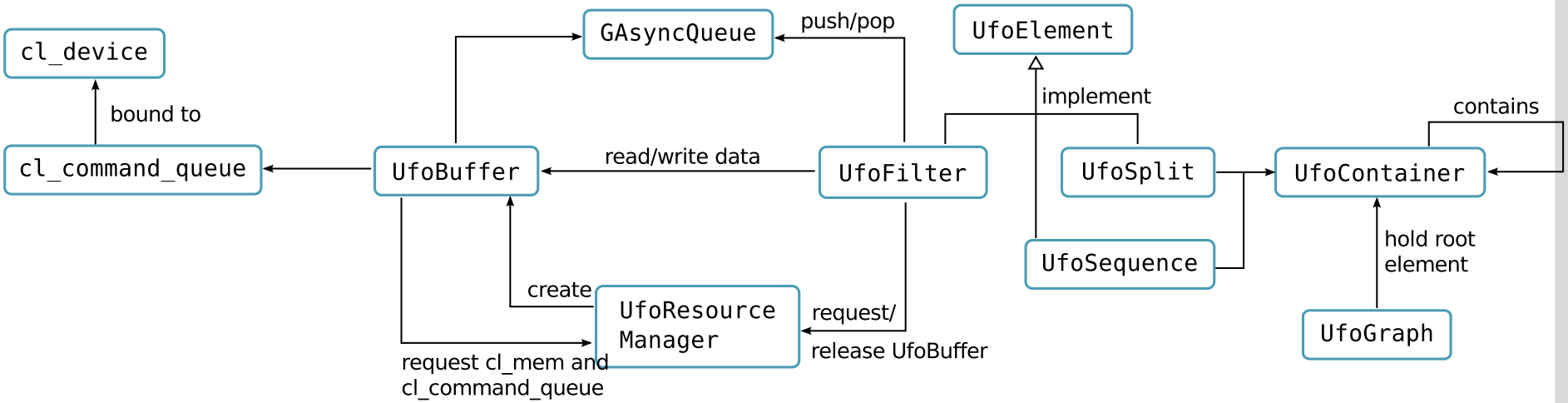
Matthias Vogelgesang – *matthias.vogelgesang@kit.edu*

Institut für Prozessdatenverarbeitung und Elektronik

Preliminaries

- *High-speed* and *low-latency* pipelining architecture for image processing
- C using GLib and GObject
  - Single inheritance
  - Interfaces
  - Signals and Properties
- OpenCL support for fast computation on GPUs
  - We should avoid unneccessary copy operations from and to GPU devices

High-Level Architecture



A Short Overview of the Documentation

Switch to browser. . .

UFO Prototype

- Basic graph and resource management is implemented
- Filters are inspected and load at run-time (aka *Plugins*)
  - Filter properties are mapped to real GObject properties
- JSON-marshalled graphs can be loaded and executed
- OpenCL and `libuca` is integrated
- CMake based

Get it here: `bzr+ssh://<user>@ufo.kit.edu/vogelgesang/ufo`

Live Demonstration

Switch to terminal. . .

Bindings

- That's nice and all, but who wants to write C?
- UFO has GObject Introspection support which means we get
  - Python,
  - JavaScript,
  - Vala
  - and more

  bindings for free
- Another small demonstration...

Remarks

- Switch from `GAsyncQueue` to `OMQ` (`zeromq.org`)?
  - Simple transition from shared memory communication to IPC and RPC
  - Would lift limitations of only $\approx$ 10 GPUs per server
  - Question: How to specify end-points, ergo which application layer protocol? (*really necessary?*)
- No arbitrary graphs possible right now, but simple to add
- How to document filter properties? (Maybe switch from Doxygen to gtk-doc?)

Conclusion

What is done?

- Simple (too simple?) but flexible architecture
- Most important integration is done already
- JSON graph description works
- Typesafe plugin mechanism works
- Language bindings are no problem

What needs to be done?

- A release strategy
- Real applications
- Graphical user interfaces (though, I'm not that sure about VisTrails)