

Visu@IGrid : A Model Driven Engineering Approach for BOINC

Christian Benjamin Ries

E-Mail: cbr@fh-bielefeld.de

Website: www.visualgrid.org

BOINC Workshop 2011, Hannover, 18th August 2011



<http://spin.fh-bielefeld.de>

SPONSORED BY THE



Federal Ministry
of Education
and Research

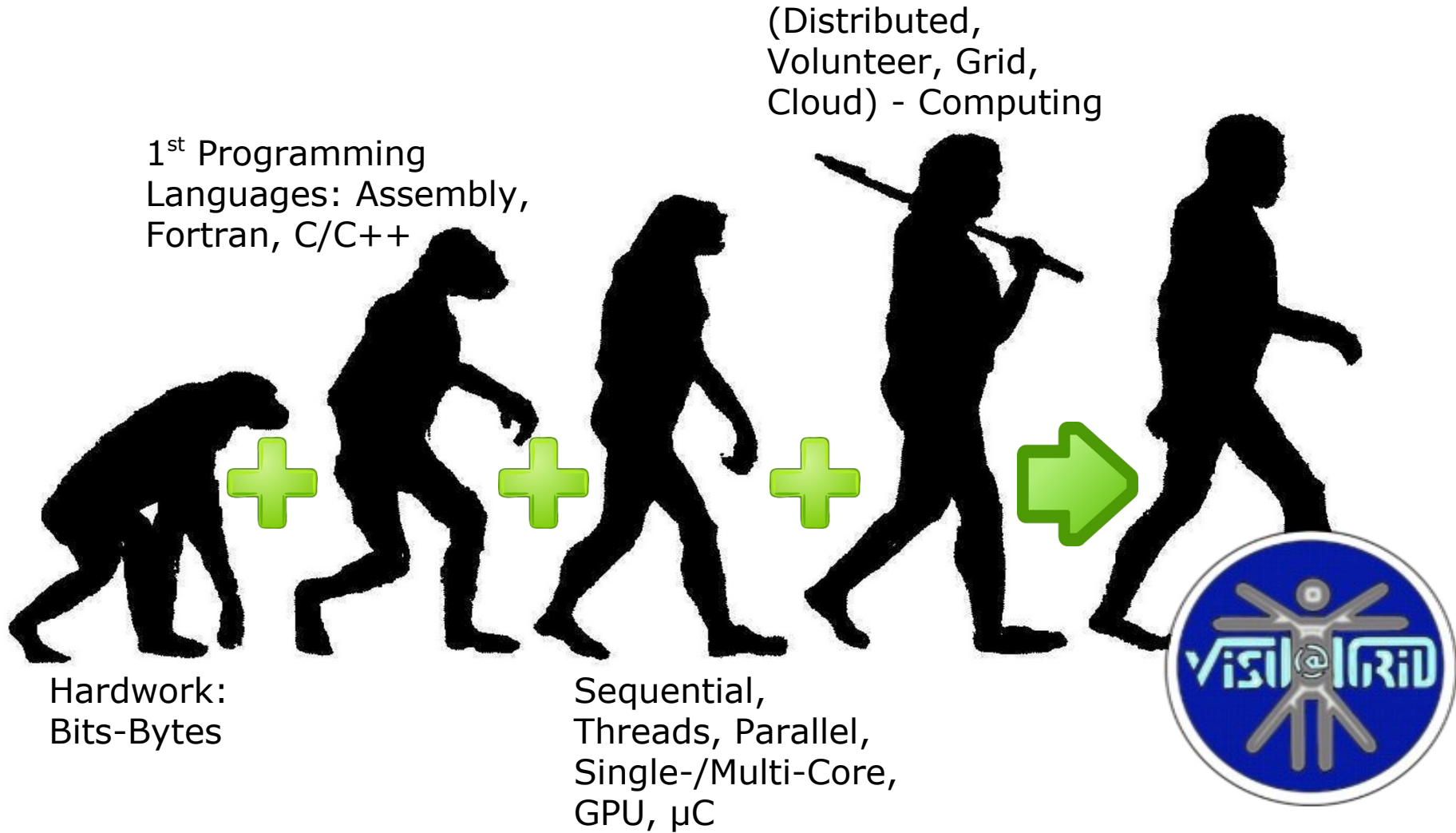


FH Bielefeld
University of
Applied Sciences

Outline

- **Introduction <<briefly>>**
- **Visu@IGrid, what happened recently, Technologies, Use-Cases**
- **Conclusion**
- **Further ideas and contributions in [close] future...**

Where are we currently?



BOINC can be used in different scenarios:

- native application [we have do everything manually],
- native application [but multithreaded, a little bit more work has to be done for thread handling],
- on different platforms and architectures,
- single-/multi-core, GPU,
- legacy applications,
- synchronous, asynchronous messages,
- ...some more...

Visu@lGrid

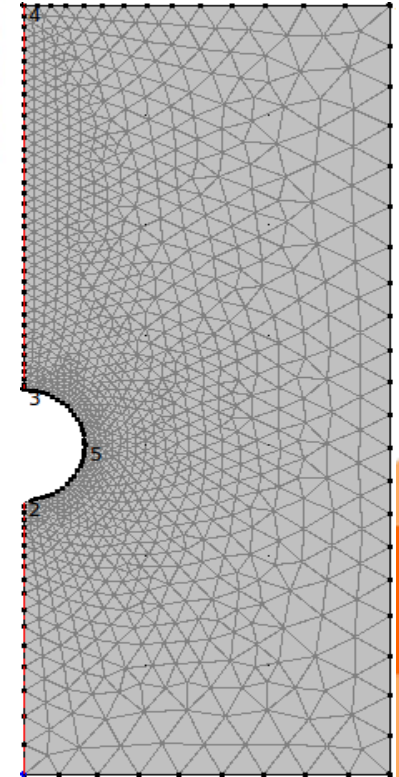
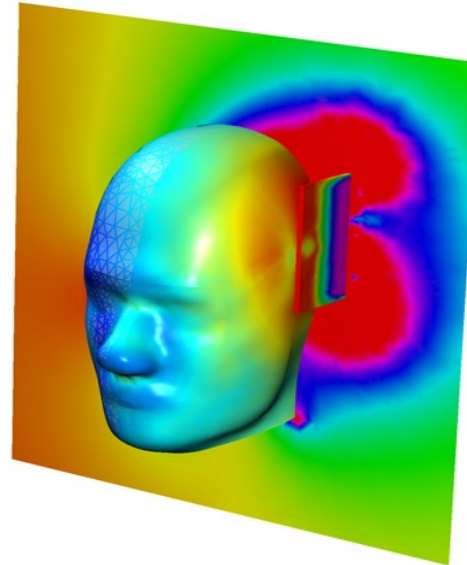
By the way...

We created a wrapper for:

- **COMSOL Multiphysics**, and

*C.B.Ries and C.Schröder, ComsolGrid - A framework for performing large-scale parameter studies using Comsol Multiphysics and Berkeley Open Infrastructure for Network Computing (BOINC), COMSOL Multiphysics Conference, **Proceedings of the COMSOL Conference**, ISBN: 978-0-9825697-6-4, France, Paris, 2010*

- **Scilab**



All scenarios need experience in one or more technology fields!

BOINC uses currently following technologies (could be extended):

- C/C++ programming language,
- Python programming language,
- BASH & CSH shell scripts,
- PHP for some monitoring, maintenance, and website elements,
- Perl (e.g. rBOINC),
- SQL (database queries),
- XML (configuration and RPC-requests),
- OpenCL & CUDA,
- ...some more!

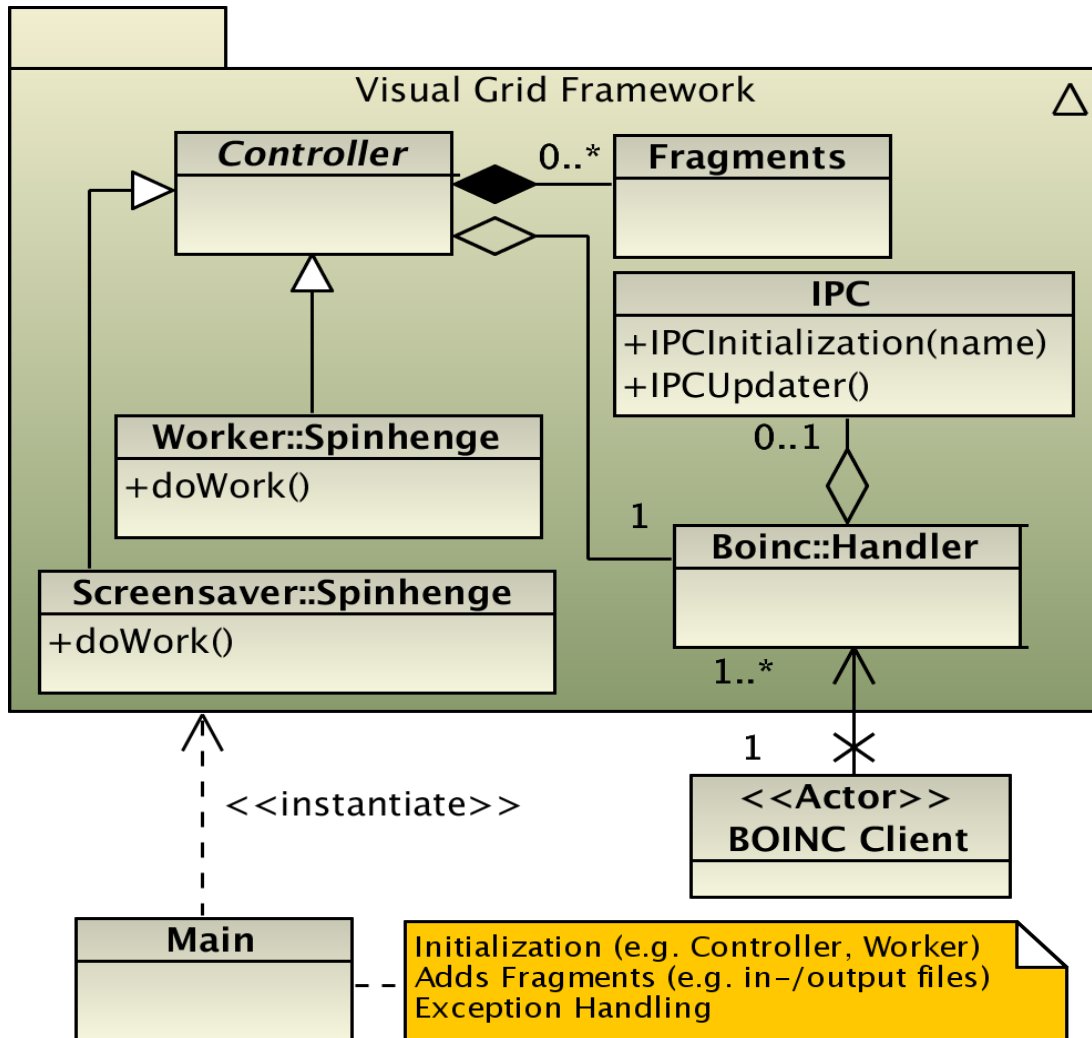
Boah... heavy...

Hard for beginners to create a BOINC project!



Visu@Grid : First step!

We created a Model-View-Controller (MVC) framework.



Here, we have created an Object-orientated framework.

Other BOINC projects did this as well.

C. B. Ries, T. Hilbig, and C. Schroeder, A Modeling Language Approach for the Abstraction of the Berkeley Open Infrastructure for Network Computing (BOINC) Framework, **Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT)**, Volume 5, IEEE, pp. 663-670, ISBN: 978-83-60810-22-4

Visu@lGrid : First step!

An example of this MVC idea:

Worker

```
class Uppercase : public VisualGrid::Controller {
private:
    VisualGrid::Int *filepos;
    inline void initVariants() {
        filepos = new VisualGrid::Int(0, "filepos");
    }

    inline void cleanVariants()
        if(filepos != NULL) delete filepos;
    }

public:
    Uppercase();
    Uppercase(const Uppercase& &u);
    ~Uppercase();

    void doWork();
    void doSharing();
};
```

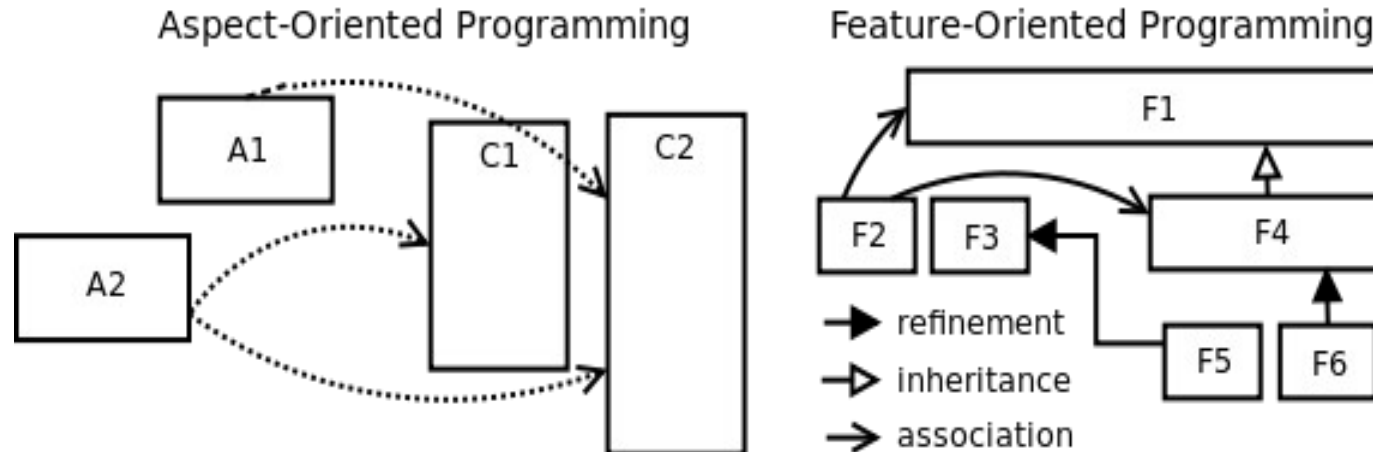
Main

```
try {
    VisualGrid::Boinc::Handler *handler =
        new VisualGrid::Boinc::Handler();
    handler->init();
    controller = new VisualGrid::Controller();

    // Files
    controller->addInputFile("in", "in");
    controller->addOutputFile("out", "out");
    controller->addHandler(handler);
    controller->start();
} catch(VisualGrid::Exceptions::Exception ex) {
    ...
}
```

Visu@lGrid : *Second step!*

We created a Model-View-Controller (MVC) framework and coupled it with an Aspect-oriented framework.

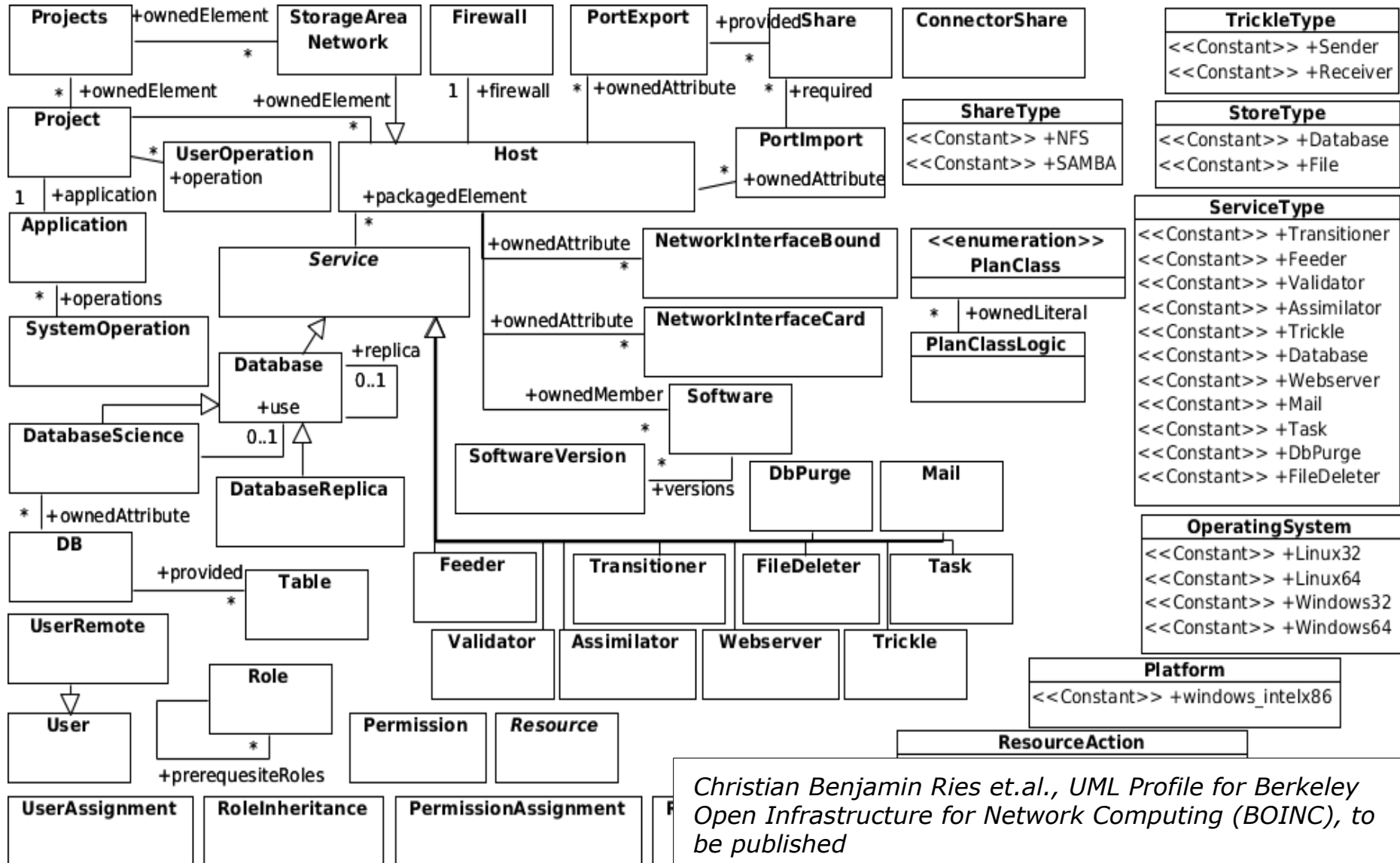


Idea & Question: Dynamic creation of scientific applications during project runtime. Any pros in relation to different configured workunits? Is it a good approach to use it within Visu@lGrid?

In progress...

Visu@Grid : Third step!

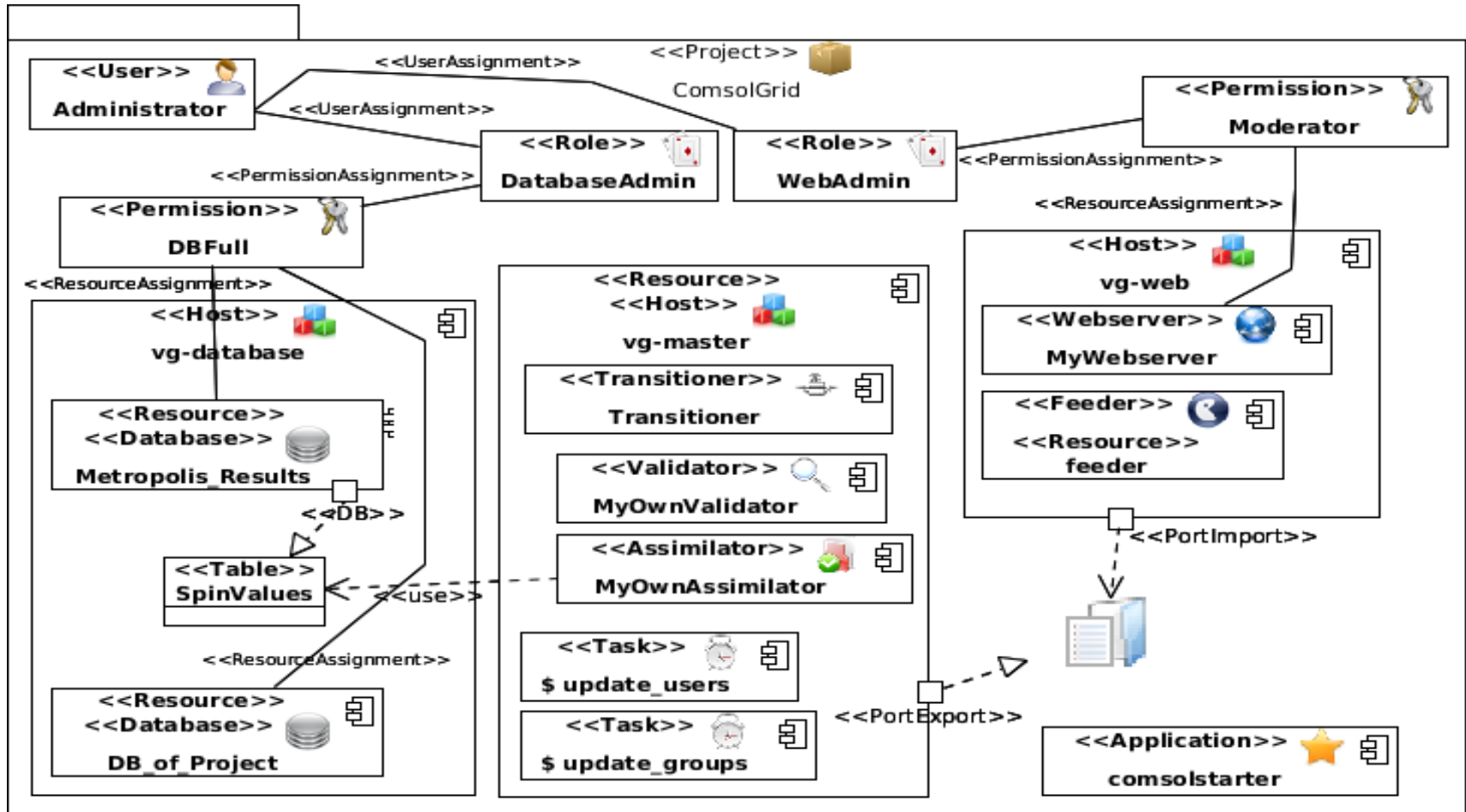
We created an UML Profile:



Christian Benjamin Ries et.al., UML Profile for Berkeley Open Infrastructure for Network Computing (BOINC), to be published

Visu@lGrid : Third step!

This UML Profile could be applied to UML diagrams:



*C.B.Ries, C.Schröder, and V.Grout, A Tree Model and an Integrated Development Environment Concept for Visu@lGrid, to be published, **Seventh Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011)***

Visu@lGrid : Forth step!

We created some Domain-specific languages, e.g.

```
base Base {
  attr name : QString;
}
```

```
Project -> Base {
  //attr projectName : QString; // Provided by Base::name.
  attr id : QString;
  attr workingDirectory : QString;
  attr state : bool;
  association operations : UserOperation [*];
  association permission : PermissionAssignment [1];
  association applications : Application [*];
  association hosts : Host [*];
  association sans : StorageAreaNetwork [*];
}
```

```
Host -> Base {
  //attr hostname : QString; // Provided by Base::name.
  attr operatingSystem : enum { Linux32, Linux64 };
  attr main : bool;
  //association firewall : Class [1];
  association permission : PermissionAssignment [1];
  association services : Service [*];
  association nics : NetworkInterfaceCards [*];
  association nicbound : NetworkInterfaceBound [*];
  association exports : PortExport [*];
  association imports : PortImport [*];
}
```

UML Definition

Deployment

```
Distributions {
  //serverLinux32: "~/ubuntu-10.1-i386.iso";
  serverLinux64: "~/ubuntu-10.1amd64.iso";
}

Project {
  maintainer: "Christian Benjamin Ries";
  contact: "cbr@fh-bielefeld.de";
  name: "visualgrid";
}
```

Science App.

```
worker Spinhenge {
  cpp {
    int a = 42;
    if(a > 42)
      modeledFunction(a);
    else a = 42;
  }
}
```

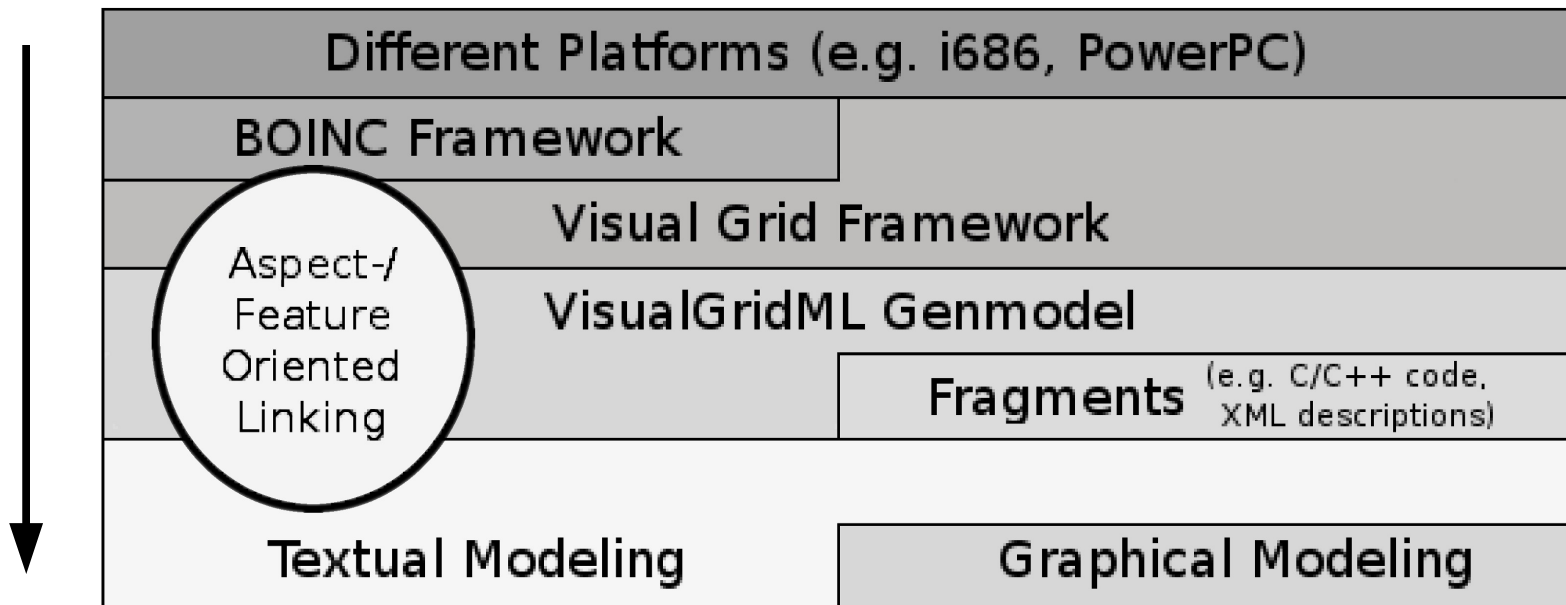
```
worker Spinhenge {
  wrapper("Matlab", "Argv[1],
    Argv[2]" [, weight, checkpointFile,
    fractionDoneFile, ...]);
}
```

Visu@Grid : Pros

Pros:

ONE MODEL -> ANY TARGET !

- Code-Generation will produce always valid and executable code!
- Platform-independent (**PIM**) and Platform-specific (**PSM**) models are definable!

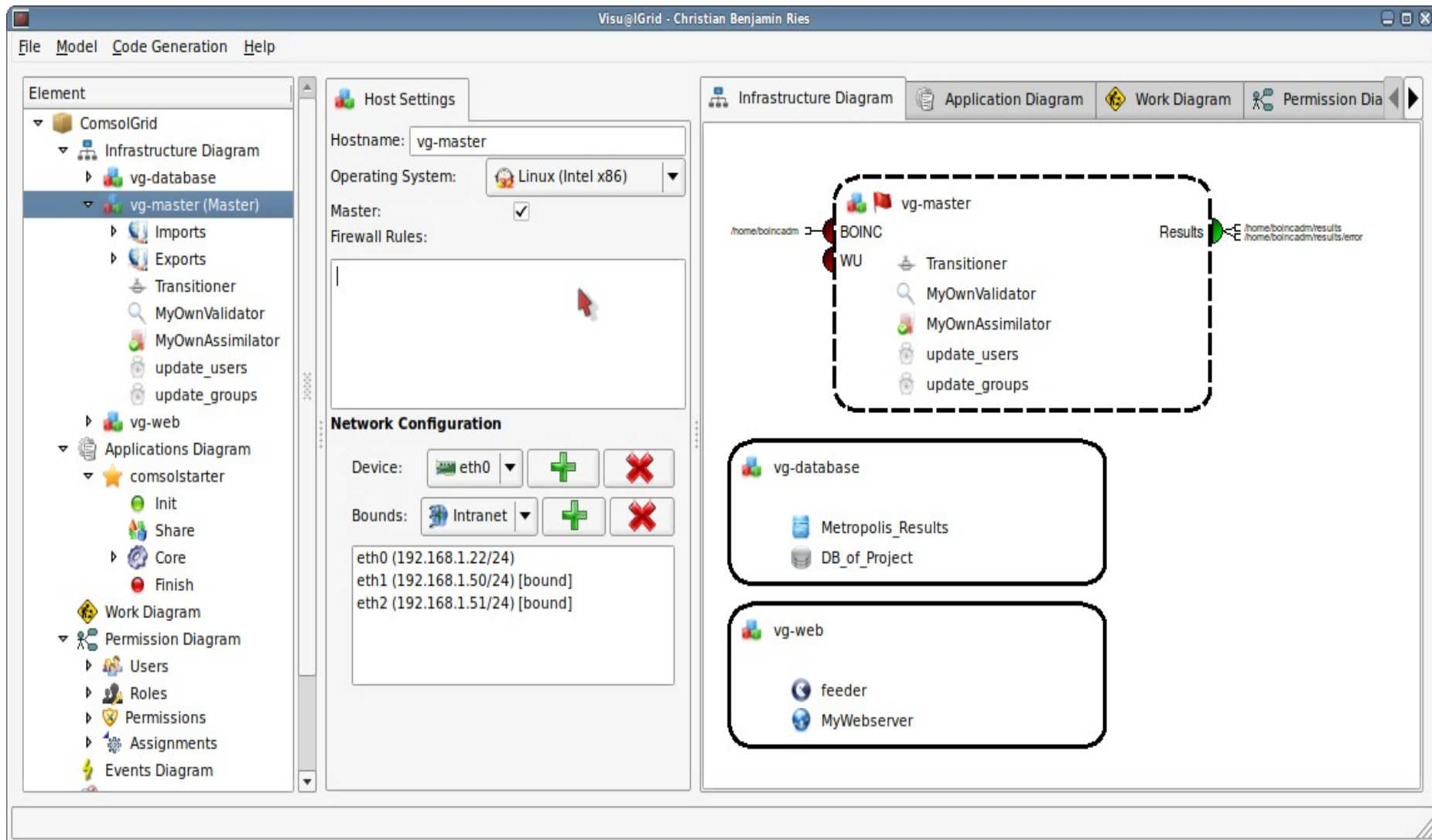


Visu@lGrid : Cons

Cons:

- Current tools are heavy and difficult to use, e.g. Eclipse.
 - > Here, I can program and love programming in C/C++, as a result I have some problems with Java and Eclipse, BUT, ~80% of work to have a MDE environment and tool facilities is done and spend for Eclipse!
- Some work have to be spend for new tools in C/C++.

Visu@IGrid : We have the time :)



The screenshot displays the Visu@IGrid software interface, titled "Visu@IGrid - Christian Benjamin Ries". The interface is divided into several sections:

- Element Tree (Left):** A hierarchical tree view showing the project structure. The "vg-master (Master)" element is selected, showing sub-elements like "Imports", "Exports", "Transitioner", "MyOwnValidator", "MyOwnAssimilator", "update_users", and "update_groups".
- Host Settings (Middle-Left):** Configuration for the "vg-master" host. Fields include:
 - Hostname: vg-master
 - Operating System: Linux (Intel x86)
 - Master:
 - Firewall Rules: (empty text area)
- Network Configuration (Bottom-Middle):** Configuration for network interfaces:
 - Device: eth0 (with + and - buttons)
 - Bounds: Intranet (with + and - buttons)
 - Interfaces: eth0 (192.168.1.22/24), eth1 (192.168.1.50/24) [bound], eth2 (192.168.1.51/24) [bound]
- Infrastructure Diagram (Right):** A diagram showing the system architecture. A dashed box encloses the "vg-master" node, which includes:
 - BOINC (with path /home/boincadm)
 - WU
 - Transitioner
 - MyOwnValidator
 - MyOwnAssimilator
 - update_users
 - update_groupsExternal connections are shown for BOINC and a "Results" node (with paths /home/boincadm/results and /home/boincadm/results/error). Below the dashed box are two solid boxes representing other components:
 - vg-database:** Metropolis_Results, DB_of_Project
 - vg-web:** feeder, MyWebserver

The Challenge...

... make it as easy as possible!

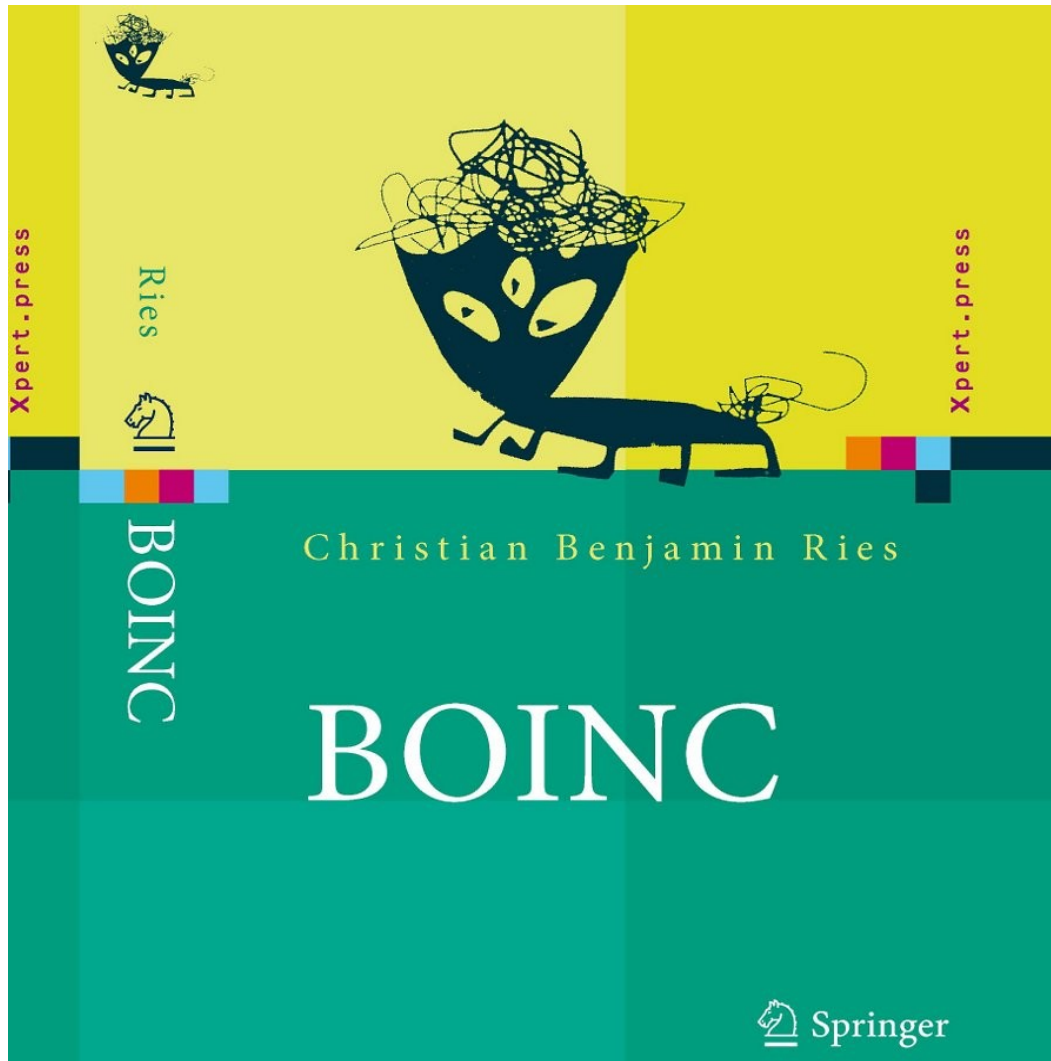
The main goal of Visu@IGrid is the realization of an **integrated development environment** which allows one to develop applications based on the „Berkeley Open Infrastructure for Network Computing (BOINC)“ by **graphical** and **textual modeling** and **complete code generation** within a **Model Driven Engineering** process!

Conclusion

- ✔ It will work!
- ✔ Visu@IGrid will be an **easy to use** way to create, deploy and maintain a BOINC based project!
- ✔ Visu@IGrid concepts can be used in additional scope of applications:
 - Message Passing Interface (MPI)
 - [hybrids] MPI and OpenMP
 - Set-up of Grid- and Cloud-Computing environments
- ✔ Open protocols allow to exchange configurations and whole BOINC projects.

Spring 2012

Your contributions are welcome!



Thank you,

Dipl.-Ing. (FH) Christian Benjamin Ries, M.Sc.

Telefon: +49 (0) 521 106 71222

e-Mail: cbr@fh-bielefeld.de

Website: www.visualgrid.org

Computational Materials Science & Engineering (CMSE)
Room 202, Werner-Bock-Straße 36
33602 D-Bielefeld



FH Bielefeld
University of
Applied Sciences