Model-based Generation of Workunits, Computation Sequences, Series and Service Interfaces for BOINC based Projects

Christian Benjamin Ries
Christian Schröder, and Vic Grout

E-Mail: mail@christianbenjaminries.de
Website: www.visualgrid.org

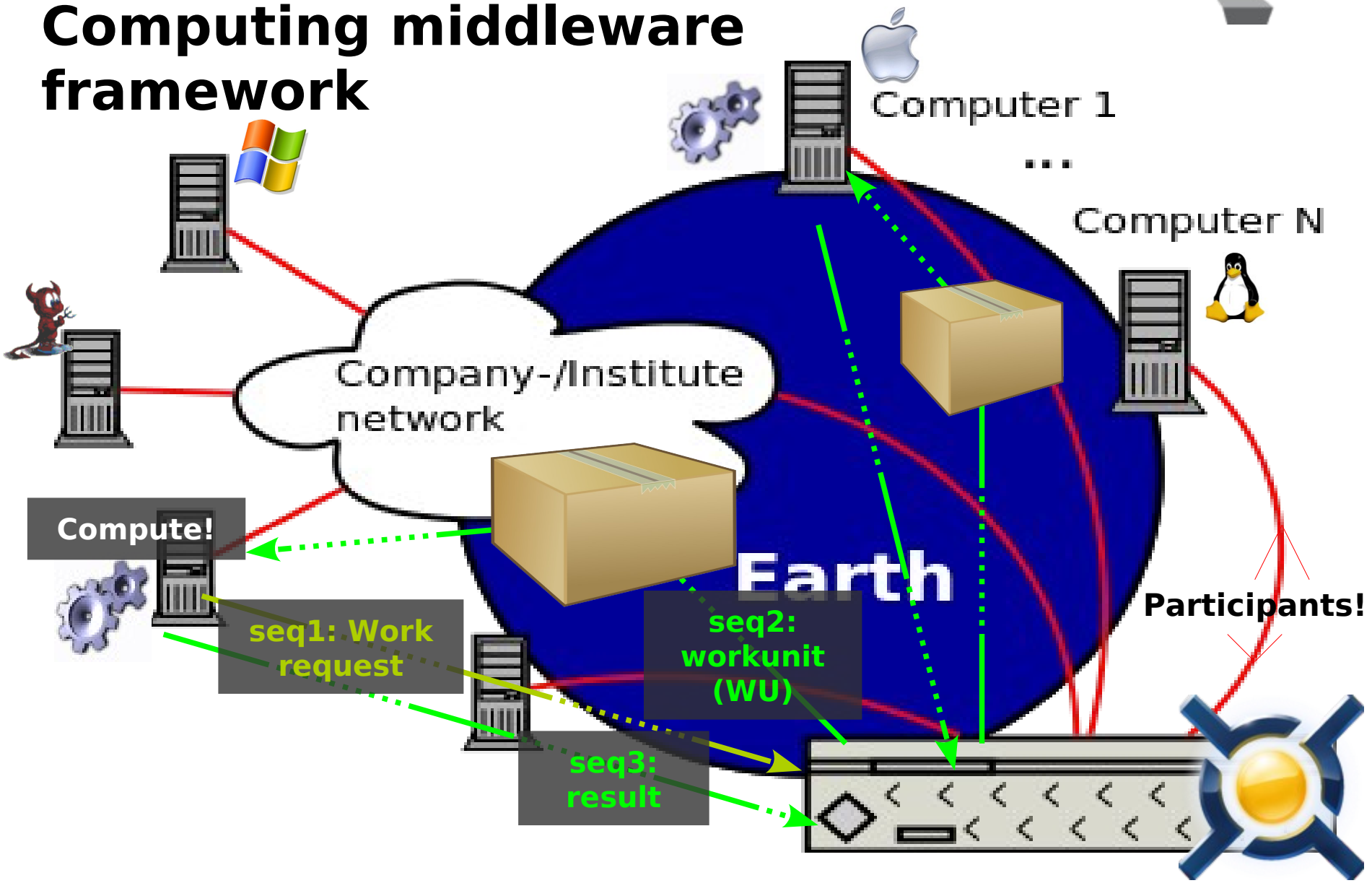
WORLDCOMP12, SERP'12
Las Vegas (USA), 17th July 2012

SPONSORED BY THE

Federal Ministry of Education and Research

FH Bielefeld
University of Applied Sciences

BOINC is a Public Resource Computing middleware framework



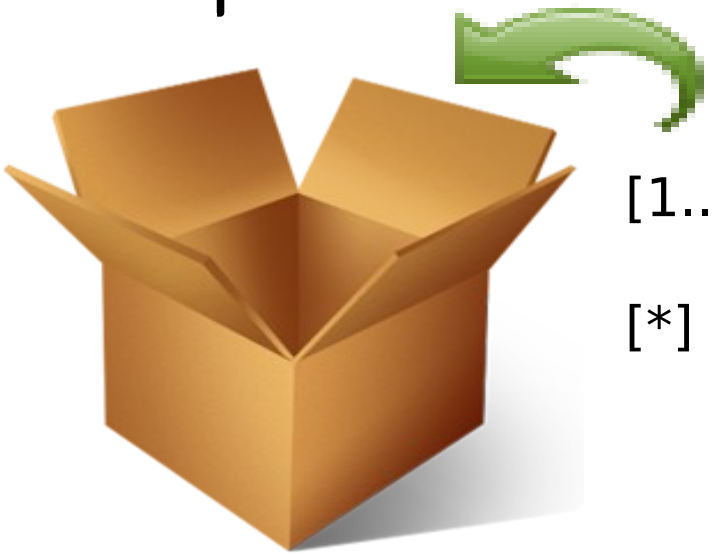
BOINC's Workunits (WUs)

FH Bielefeld
University of
Applied Sciences

[1]

Scientific Application
(Independent Software Vendor, Legacy-applications, personal creative implementations...)

[0..*]

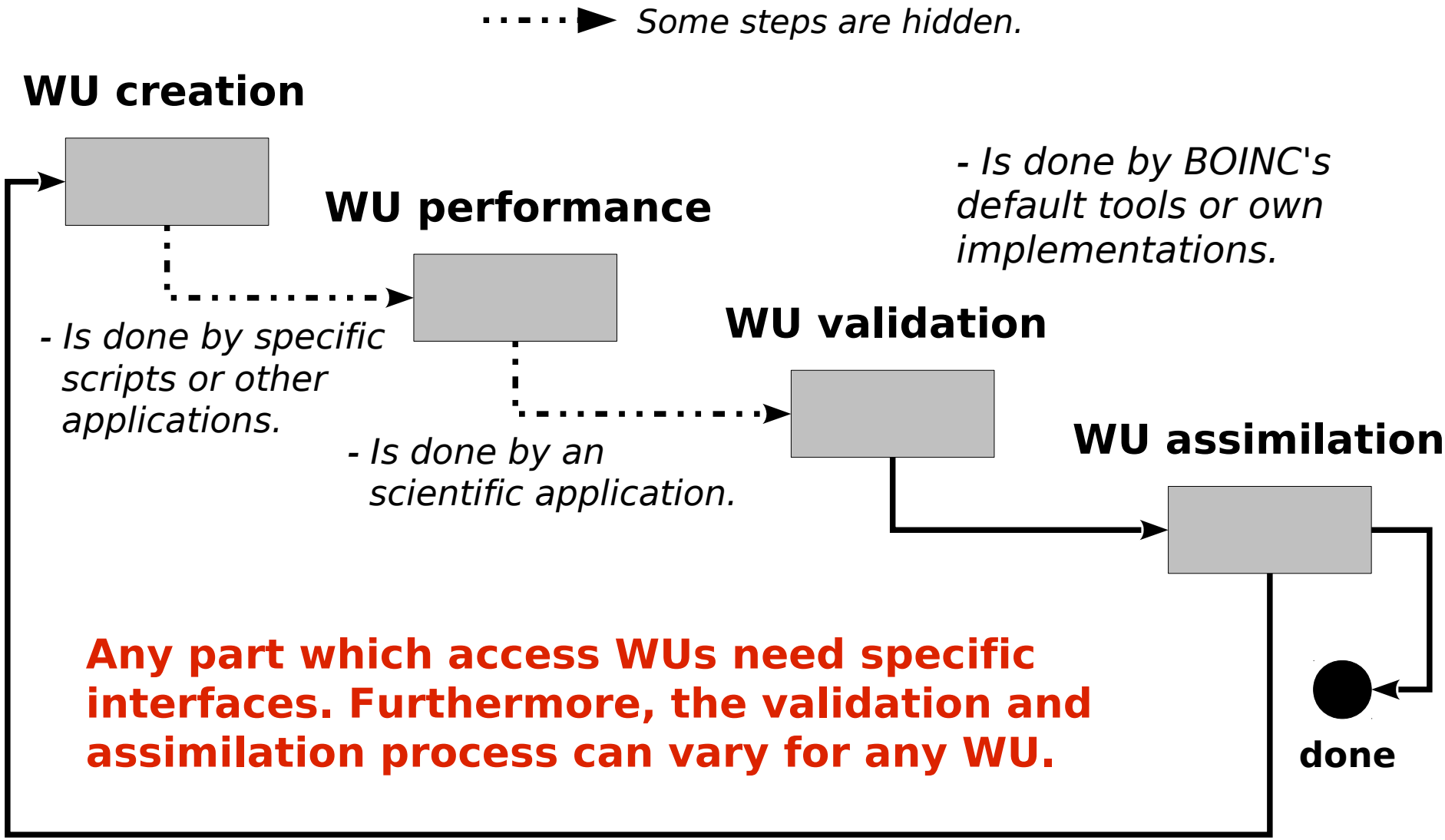


[1..*] various sets of runtime-parameter

[*] additional files

Excerpt of BOINC's WU Lifetime

FH Bielefeld
University of
Applied Sciences

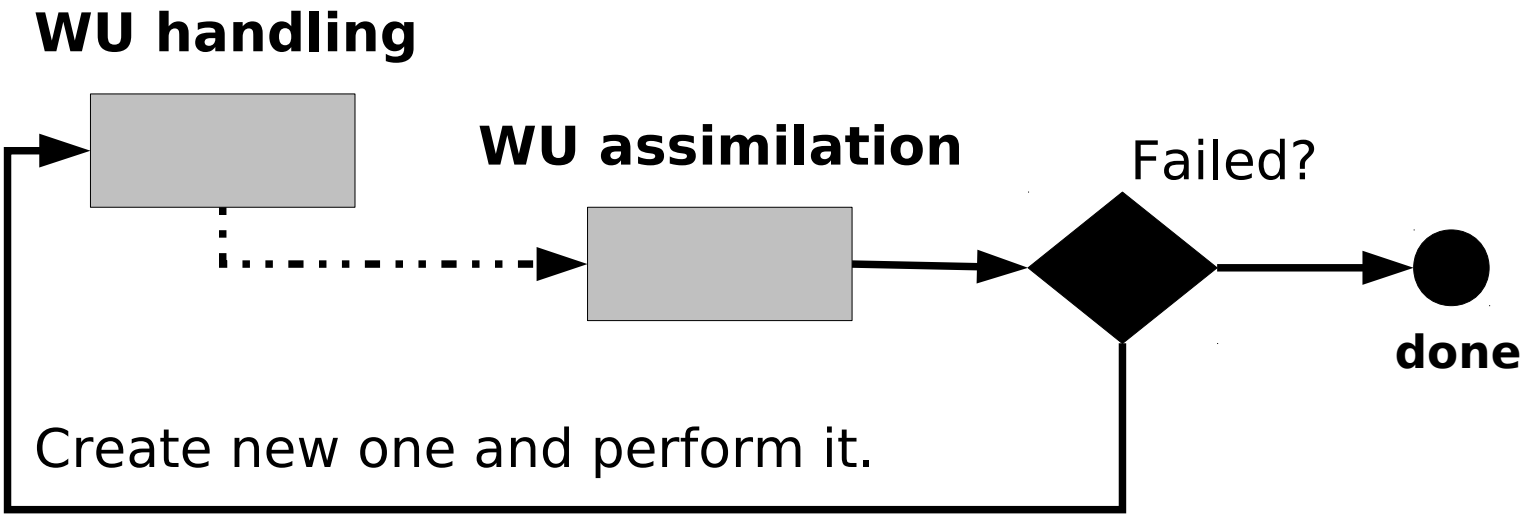


This transition is on the next slide.

Handling of failed WUs

FH Bielefeld
University of
Applied Sciences

Some steps are hidden.



Handling of WU Series

FH Bielefeld
University of
Applied Sciences

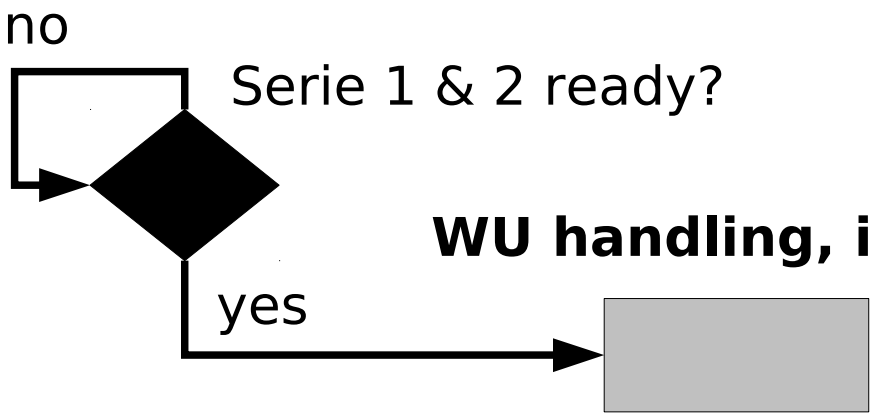
WU handling, id: 10

Serie 1:

WU handling, id: 100

Serie 2:

Serie 3:



WU handling, id: 110

Uses the result values of WUs (id: 10 and id: 100).

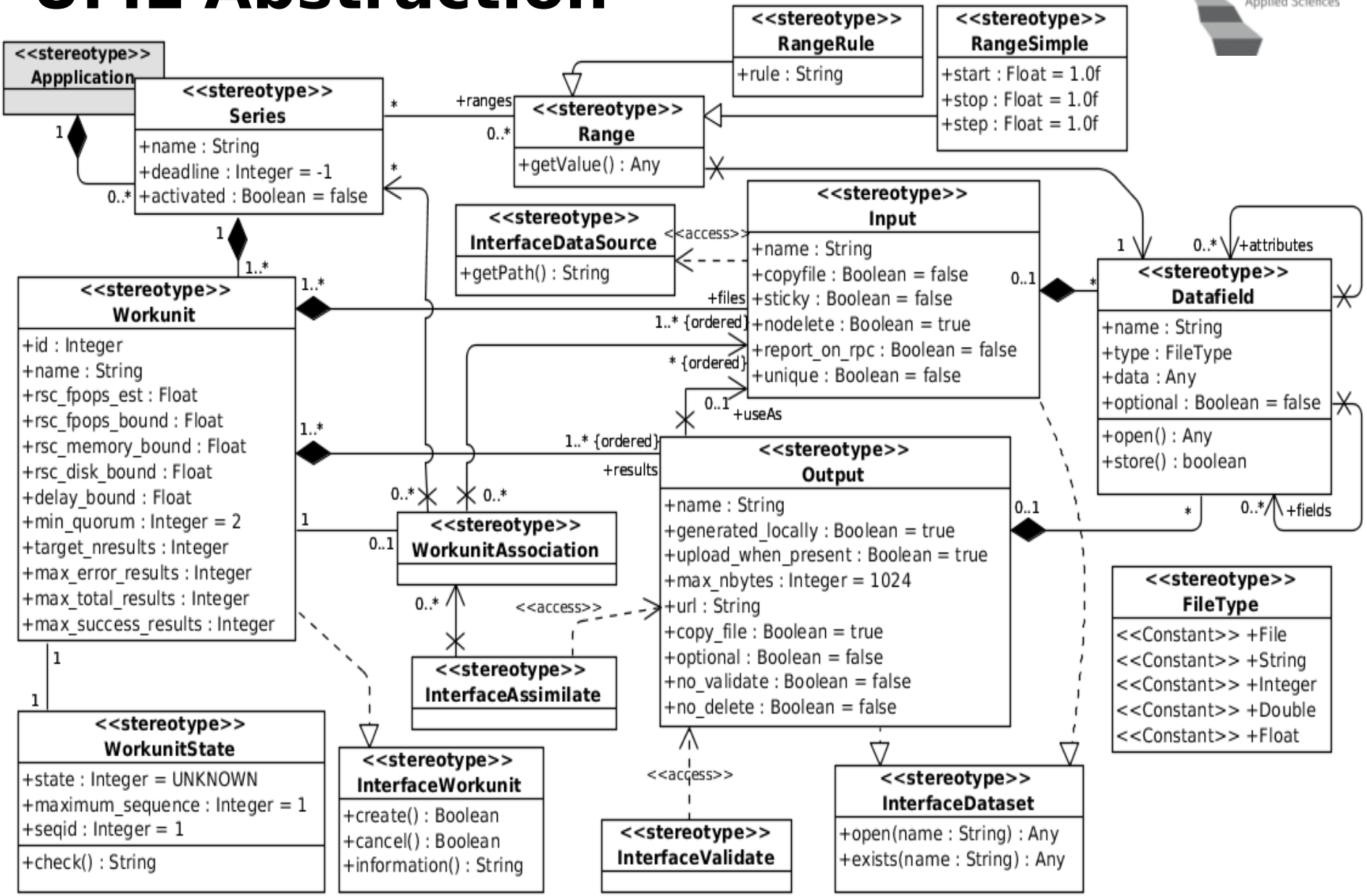
The Core Questions

FH Bielefeld
University of
Applied Sciences

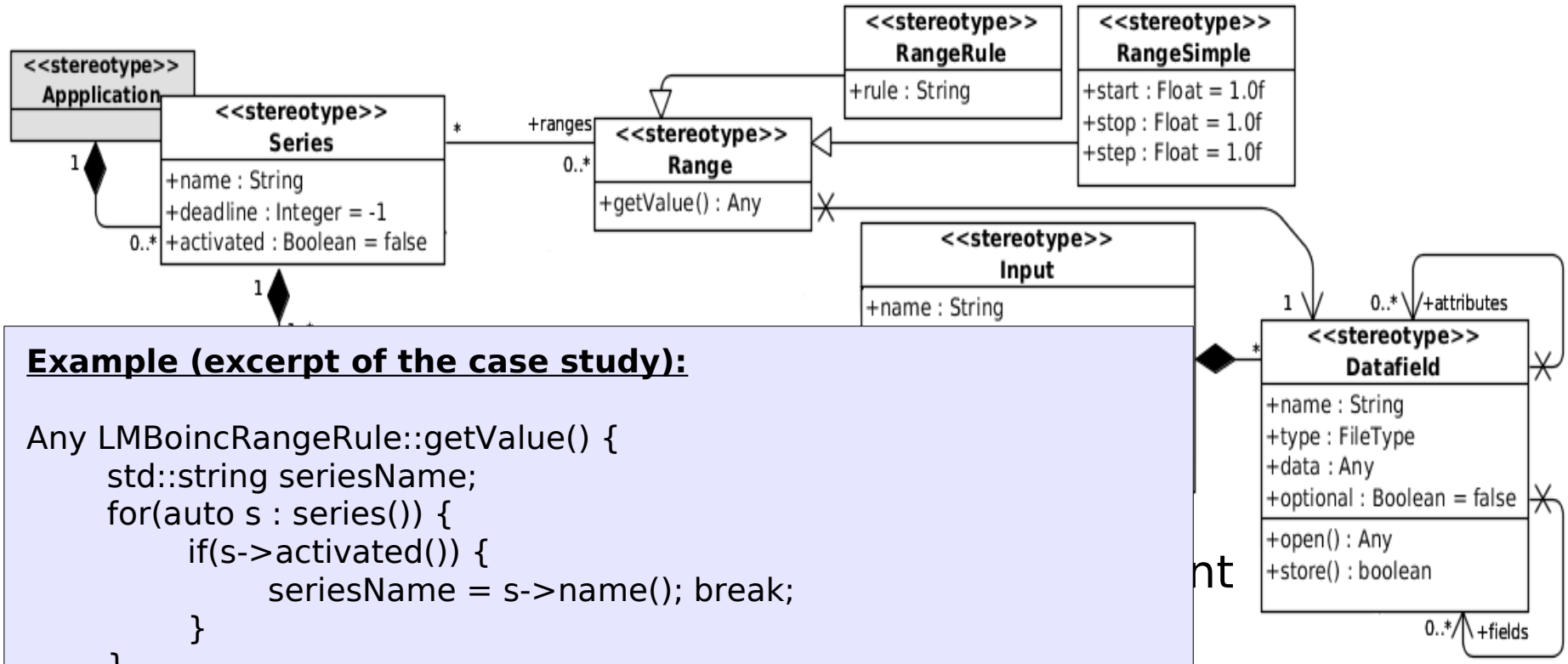
- Which UML elements are necessary to create a model for WU creation?
- How can we model a sequential queue for WU progressing?
- How can BOINC's validator and assimilator access result's data on a higher abstraction level? In addition, is it possible to have only one interface or description which makes it possible to allow access by all BOINC components?
- How can we track the lifetime of WUs when they are used in different scenarios, e.g. one WU is used within a sequential performed queue?

UML Abstraction

FH Bielefeld
University of
Applied Sciences



UML Abstraction: Ranges

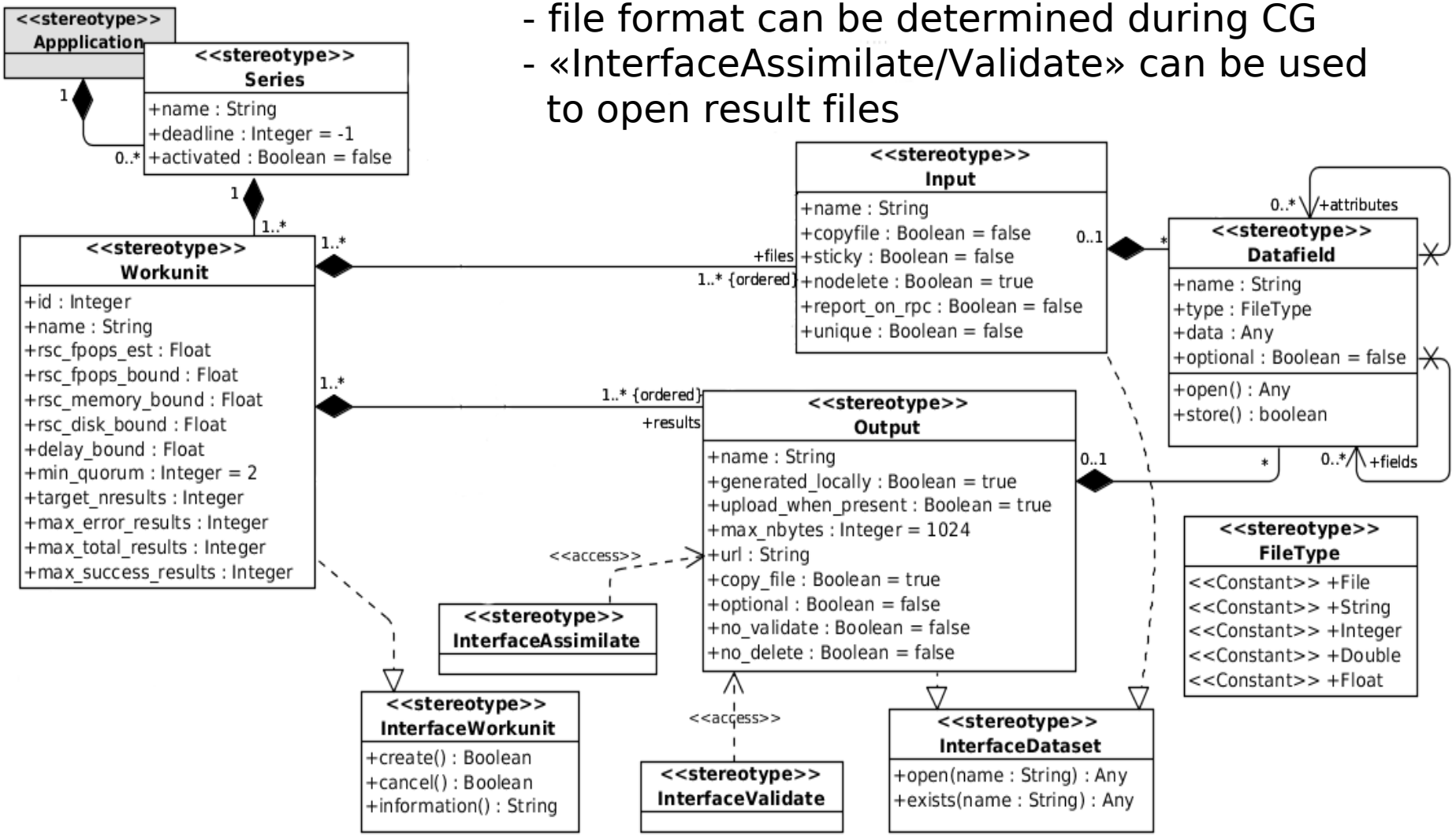


Example (excerpt of the case study):

```
Any LMBoincRangeRule::getValue() {
      std::string seriesName;
      for(auto s : series()) {
            if(s->activated()) {
                  seriesName = s->name(); break;
            }
      }
           if(seriesName == "S1")    datafield()->setData("edge");
      else if(seriesName == "S2")    datafield()->setData("normalize");
      else if(seriesName == "S3")    datafield()->setData("oil");
      else if(seriesName == "S4")    datafield()->setData("negate");
      else if(seriesName == "S5")    datafield()->setData("combine");
      return datafield()->data();
}
```

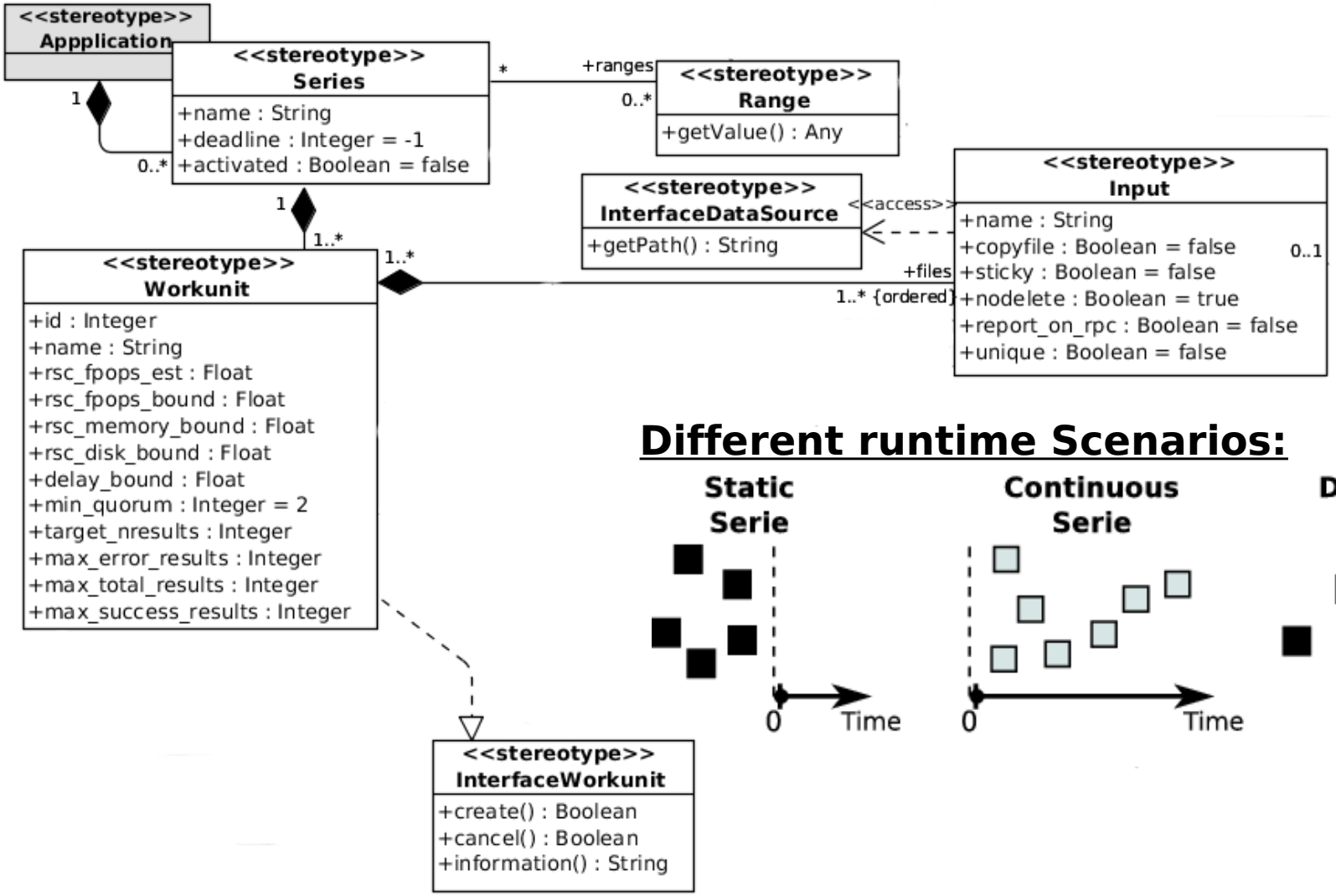
UML Abstraction: In-/Output, Datafields

- file type can be queried during CG
- file format can be determined during CG
- «InterfaceAssimilate/Validate» can be used to open result files

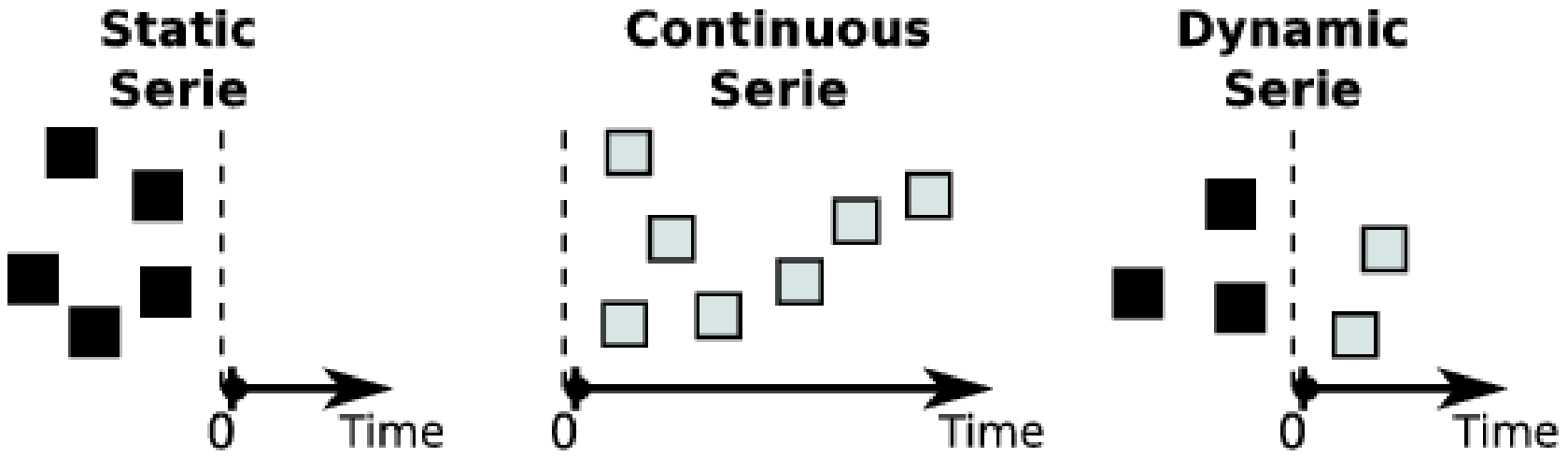


UML Abstraction: WUs on Demand

FH Bielefeld
University of
Applied Sciences



Handling of Series

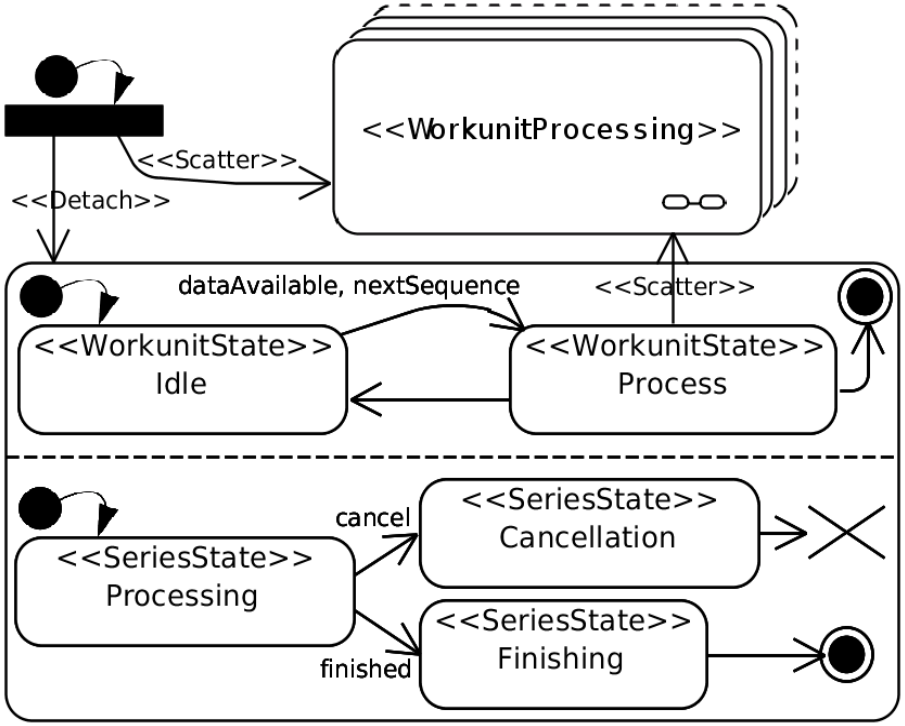


- **Static:** All WUs are created before a «Series» will be created. Only these known WUs are handled by a BOINC project.

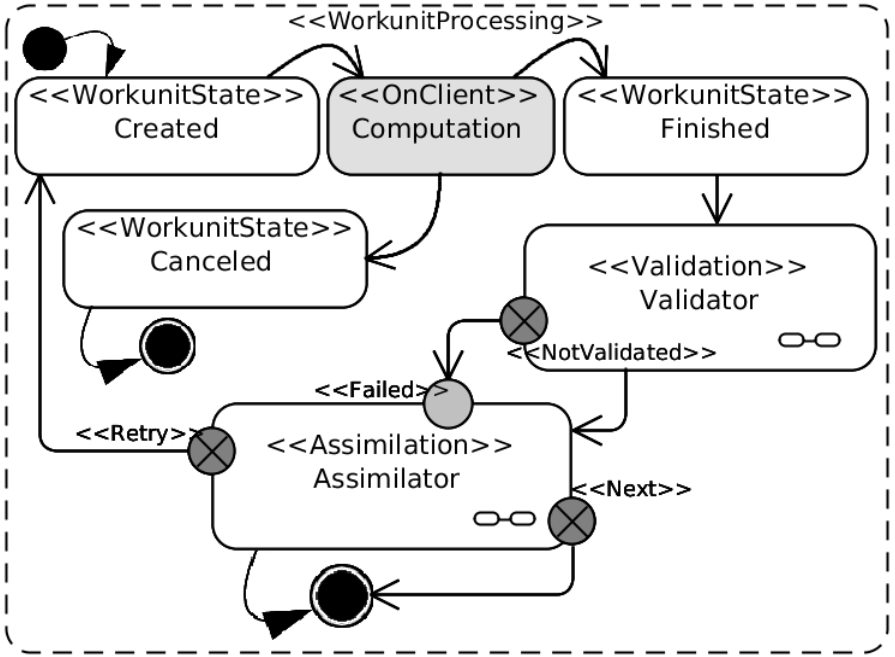
- **Continuous:** No WUs at the beginning of a «Series». WUs are created on demand, e.g. when new data packages are available or when a time slot is reached.

- **Dynamic:** The previous two possibilities are merged.

Series & WU States



First part of the UML statemachine diagram to monitor instances of «Workunit» and «Series». State's top region is responsible for WU monitoring and creates new WUs when data is available or next WU in a sequence has to be performed. The bottom region monitors a «Series» and handles canceling events for a «Series» instance or if its finished and results can be merged.

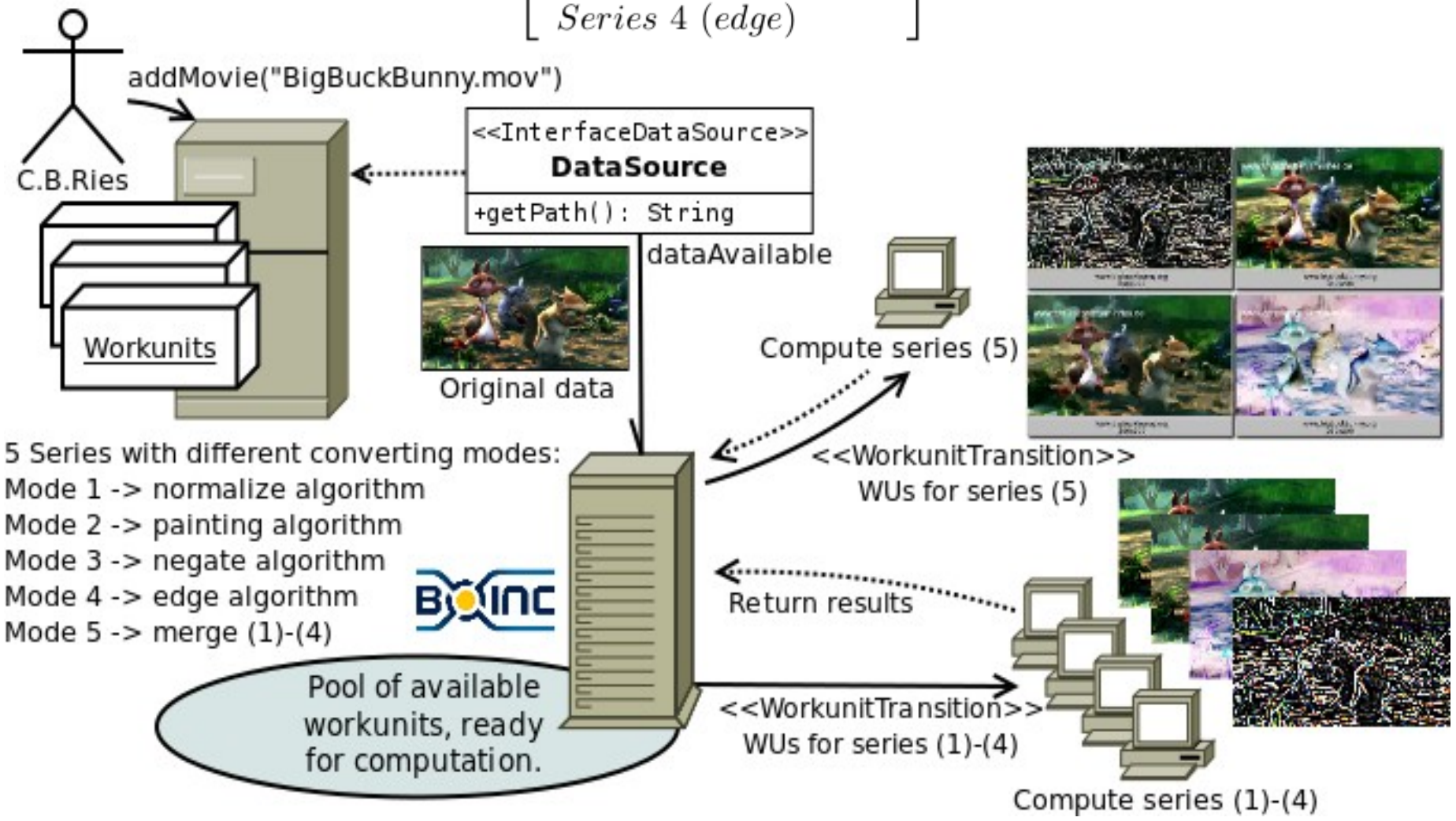


Second part of the UML statemachine diagram. During computation of one WU, clients can decide to cancel current WU, and therefore it has a changed WU state. When it is finished it will be validated, if this validation failed the exit pseudostate is used. The followed assimilation state can decide to retry this WU and a new WU is created with same «Input» values. If this WU is in a sequence, "Next" is used otherwise the statemachine is finished.

Case Study

5 Series:

$$\begin{bmatrix} Series\ 1\ (normalize) \\ Series\ 2\ (painting) \\ Series\ 3\ (negate) \\ Series\ 4\ (edge) \end{bmatrix} \Rightarrow Series\ 5\ (merge)$$



Future Work

FH Bielefeld
University of
Applied Sciences

- WU's performance can have restrictions, e.g. the use of floating-point operations or allocation of hard disk space can be limited. Is it possible or reasonable to use the UML for this kind of specification?

- The current case study is hard-coded and cannot be changed during runtime. This can be changed by the use of a Domain-specific language (DSL) or by the use of a graphical language.

- Additional thought should also be spent on how the presented model can be used for conventional supercomputers, where other technologies like Message Parsing Interface (MPI) or OpenMP are used.

Thank you,

Christian Benjamin Ries

e-mail: mail@christianbenjaminries.de
website: www.visualgrid.org

FH Bielefeld
University of
Applied Sciences