Code Generation Approaches for an Automatic Transformation of the Unified Modeling Language to the BOINC Framework

Christian Benjamin Ries
and Vic Grout

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

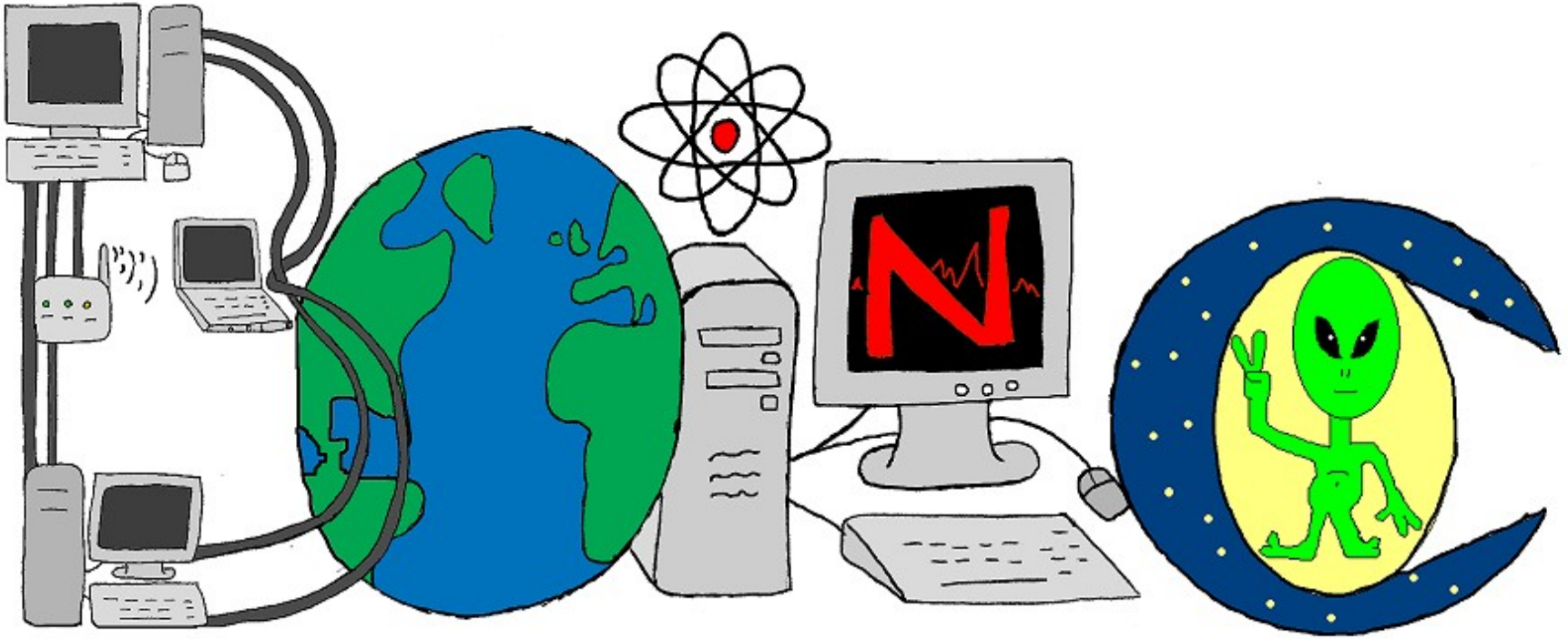
SCSE‘13
San Francisco (USA), 1st March 2013

SPONSORED BY THE

Federal Ministry
of Education
and Research

Overview

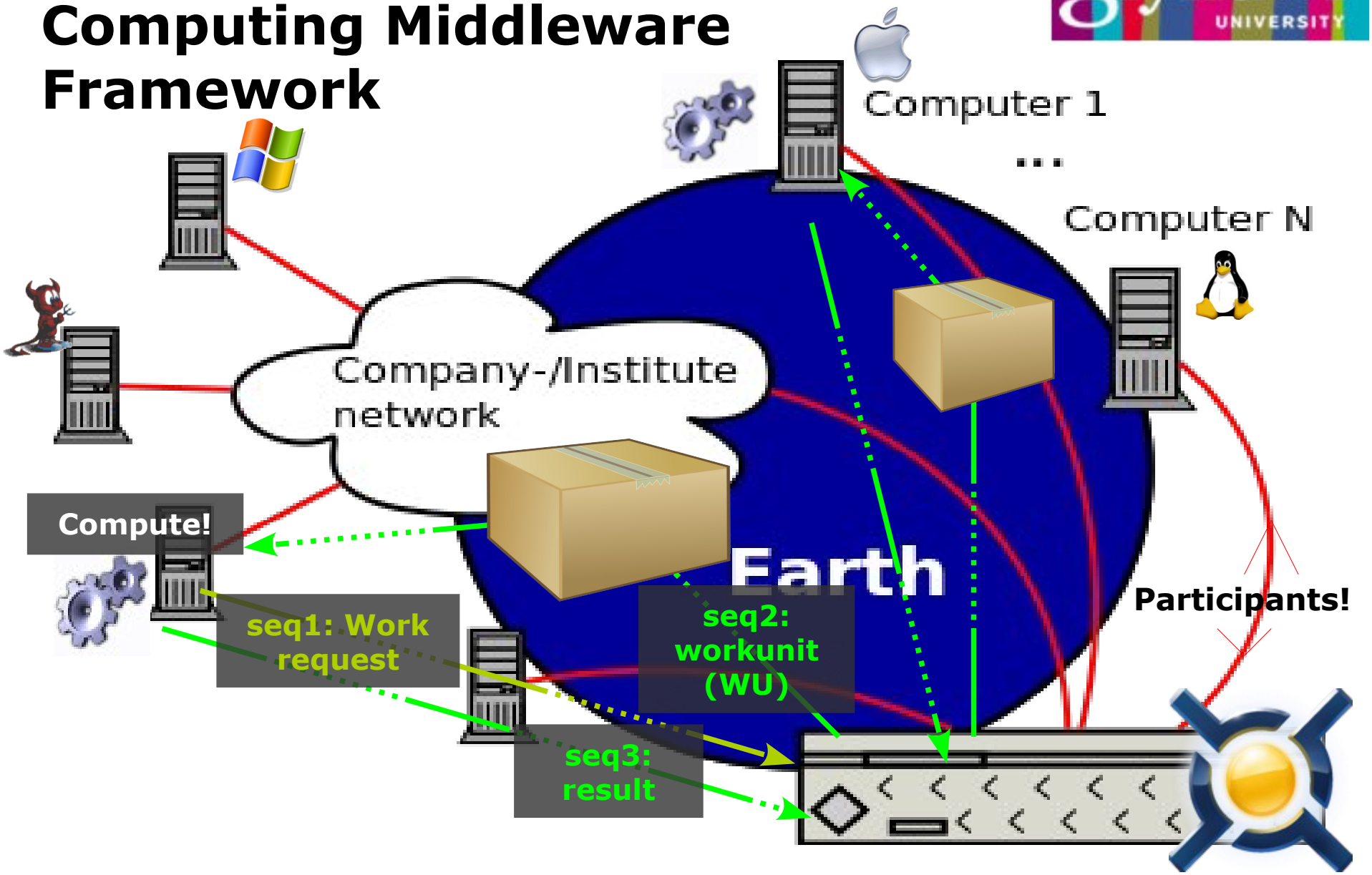
- Briefly Introduction of **BOINC**
- Briefly Introduction of **Code Generation (CG)**
- A selection of **CG approaches**



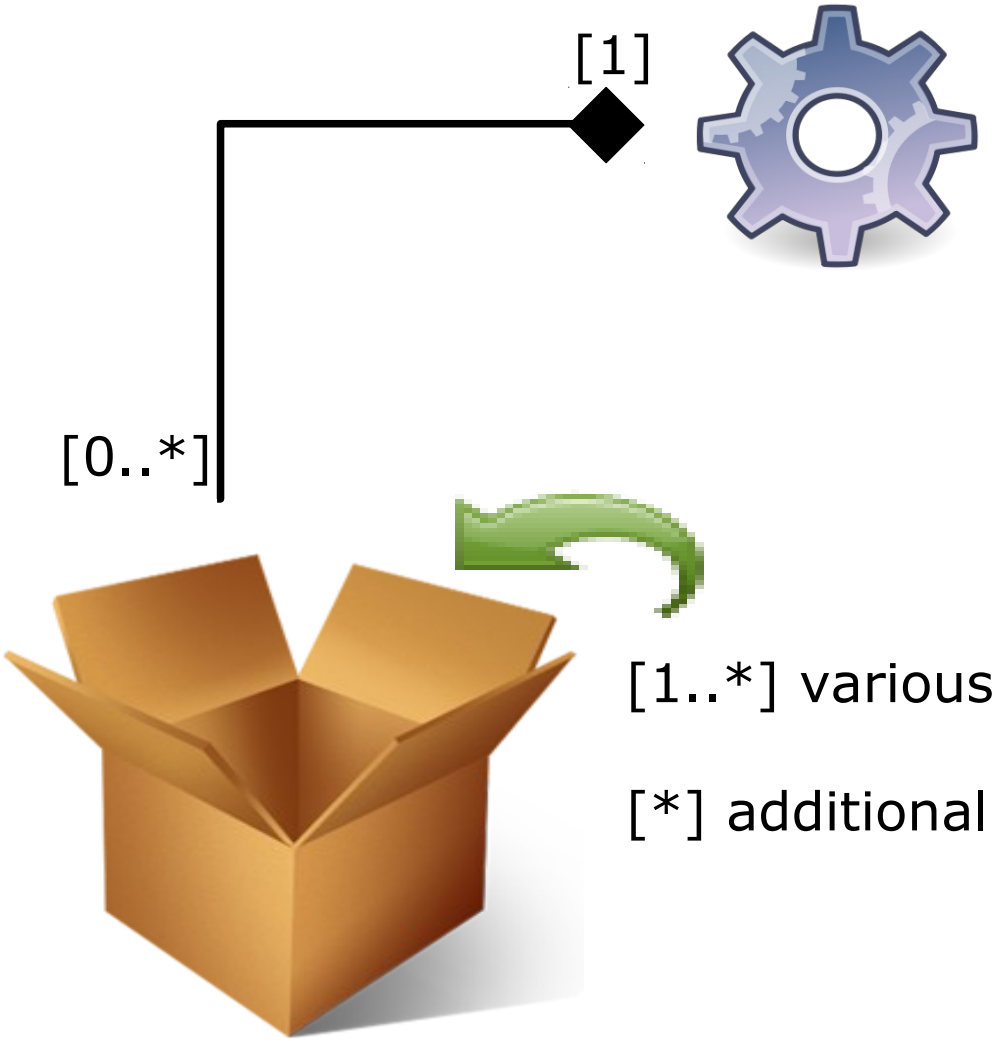
Berkeley Open Infrastructure for Network Computing

Image source:
http://www.seti.cl/aprendiendo-mas-sobre-boinc-y-la-computacion-voluntaria/

BOINC is a Public Resource Computing Middleware Framework



BOINC's Workunits (WUs)

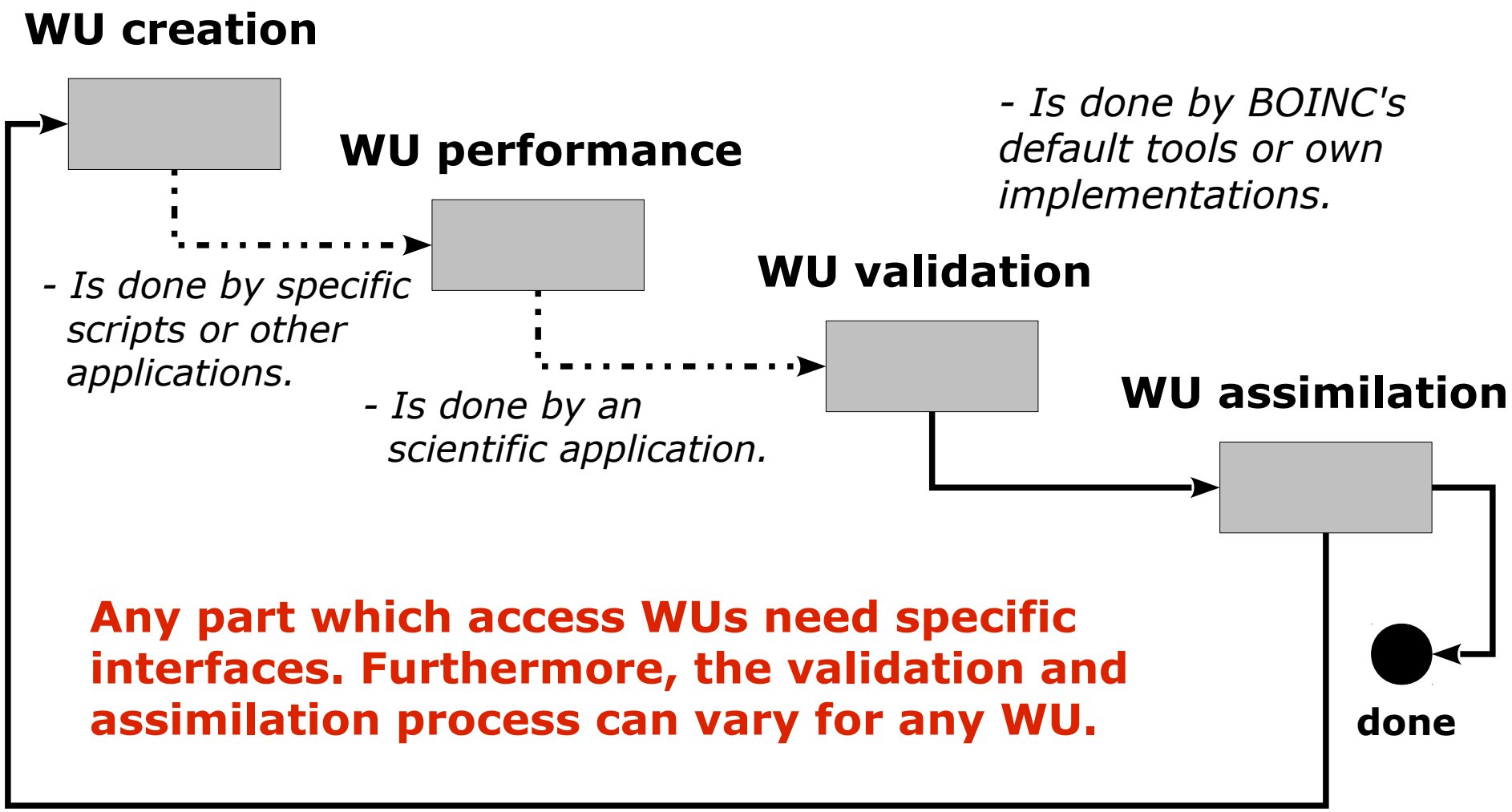


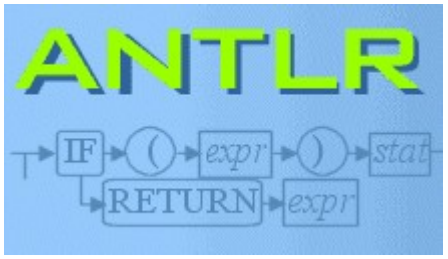
Scientific Application
(Independent Software Vendor, Legacy-applications, personal creative implementations...)

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

[*] additional files

Excerpt of BOINC's WU Lifetime



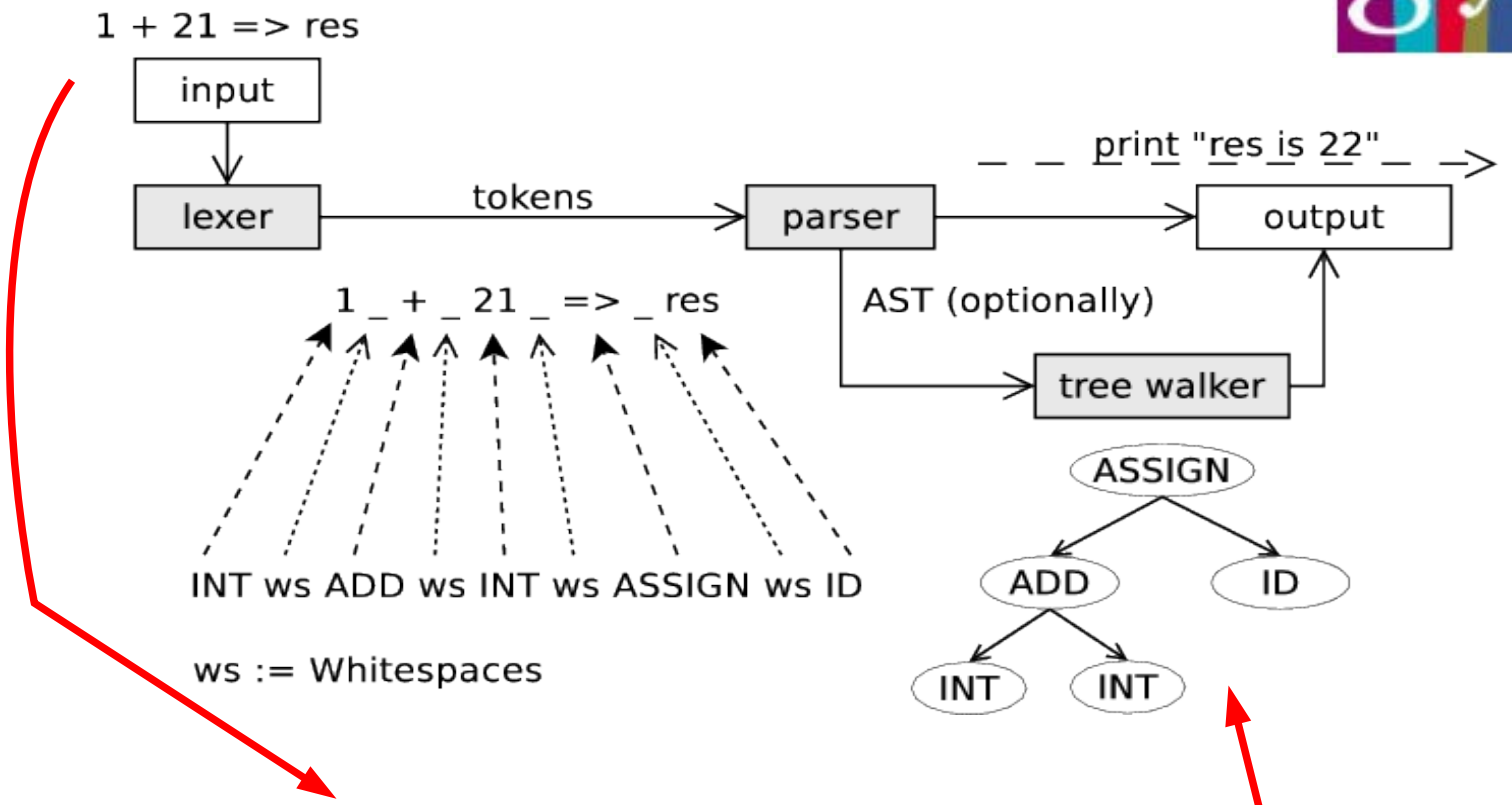


ANother **T**ool for **L**anguage **R**ecognition

http://www.antlr.org

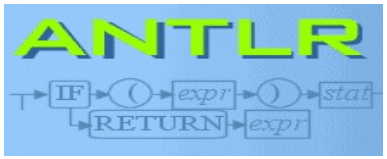
Current version is 4!

ANTLR's Abstract-syntax Tree




```
summation : leaf* -> leaf* ;
leaf : INT ('+' INT)* '=>' ID ->
    ^(ASSIGN ^(ADD INT*) ID);
```

Listing 2. Definition of the AST of the summation.



+ BOINC +

UNIFIED MODELING LANGUAGE

glyndŵr UNIVERSITY

(1) DSML for semantic-model creation

<<generates>>

<<uses>>

(2) Semantic-model libraries (C++)

<<uses>>

(6) Model

(3) DSML

(4) C++-Model

(5) Visual-Model

Includes: (7) libries
- Generators
- Generator rulesets
- Datatype definitions
- Helper functions

<<uses>>

<<creates>>

1) BOINC-Configurations
 1.1) Server-components
 1.2) Services
 1.3) Shares and Mounts
 1.4) Tasks
2) Service/Task implementations
3) Access-control lists
4) Scientific application
 4.1) implementation
 4.2) deployment
5) Workunits
 5.1) Creation
 5.2) Submittion
 5.3) Handling
6) (opt) Asynchronous-messages
7) (opt) Monitoring
8) (opt) Exception handling

Code Generation Approaches

Work Validation

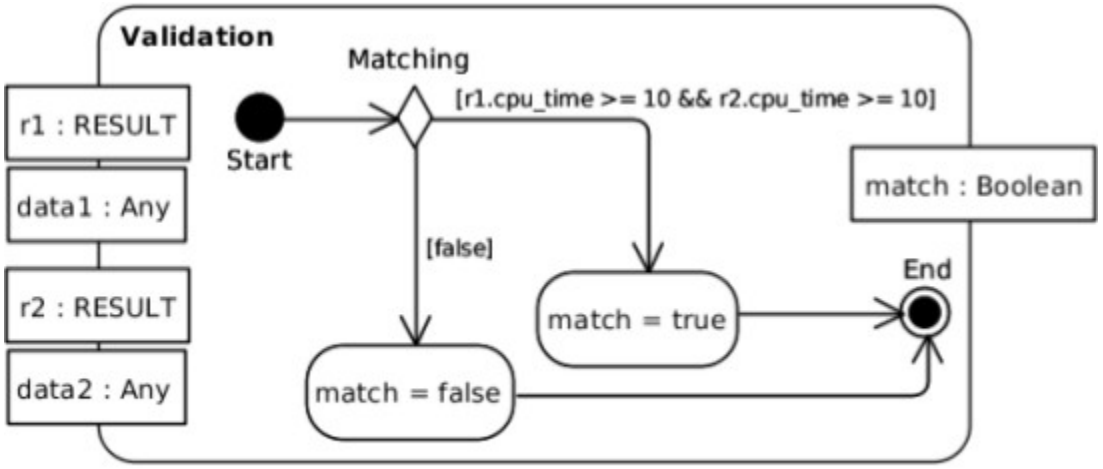


Figure 16. Part of BOINC's high-level validation framework in UML.

Transformation to an AST; followed by a code generation process (included in source).

```
int compare_results(RESULT & r1, void *data1,
                    RESULT & r2, void *data2,
                    bool & match) {
 match = (r1.cpu_time >= 10 && r2.cpu_time >= 10);
 return 0;
}
```

Listing 13. Part of BOINC's high-level validation framework.

Code Generation Approaches

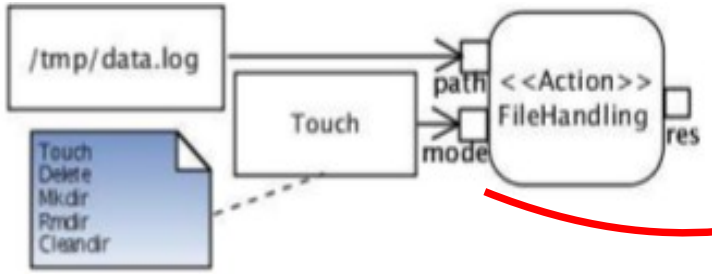


Figure 10. «Action» for file handling with two parameters.

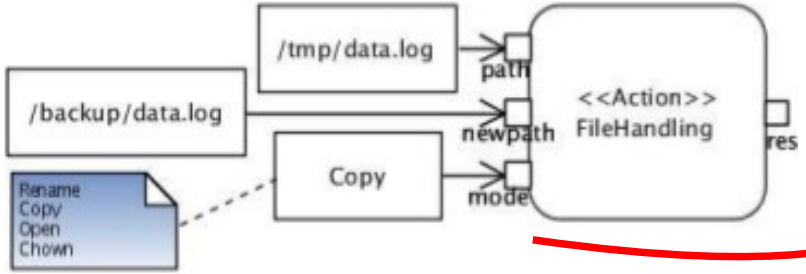


Figure 11. «Action» for file handling with three parameters.

Mode	BOINC's Function Call
one additional input pin	
Touch	int res = boinc_touch_file (path);
Delete	int res = boinc_delete_file (path);
Mkdir	int res = boinc_mkdir(path);
Rmdir	int res = boinc_rmdir(path);
Cleandir	int res = clean_out_dir (path);
two additional input pins	
Copy	int res = boinc_copy(path, newpath);
Chown	int res = boinc_chown(path, owner);
Rename	int res = boinc_rename(path, newpath);
Open	FILE *res = boinc_fopen(path, filemode);

Table II
MAPPING OF UML TO BOINC'S API.

- Easier recognition of functionality
- UML stereotypes can be replaced by specific icons (i.e. similar to Matlab or Simulink).

Code Generation Approaches

Client-side

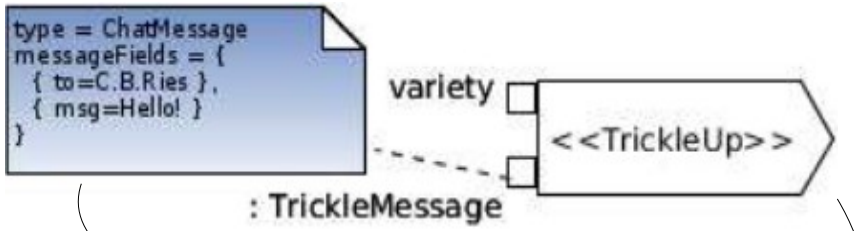
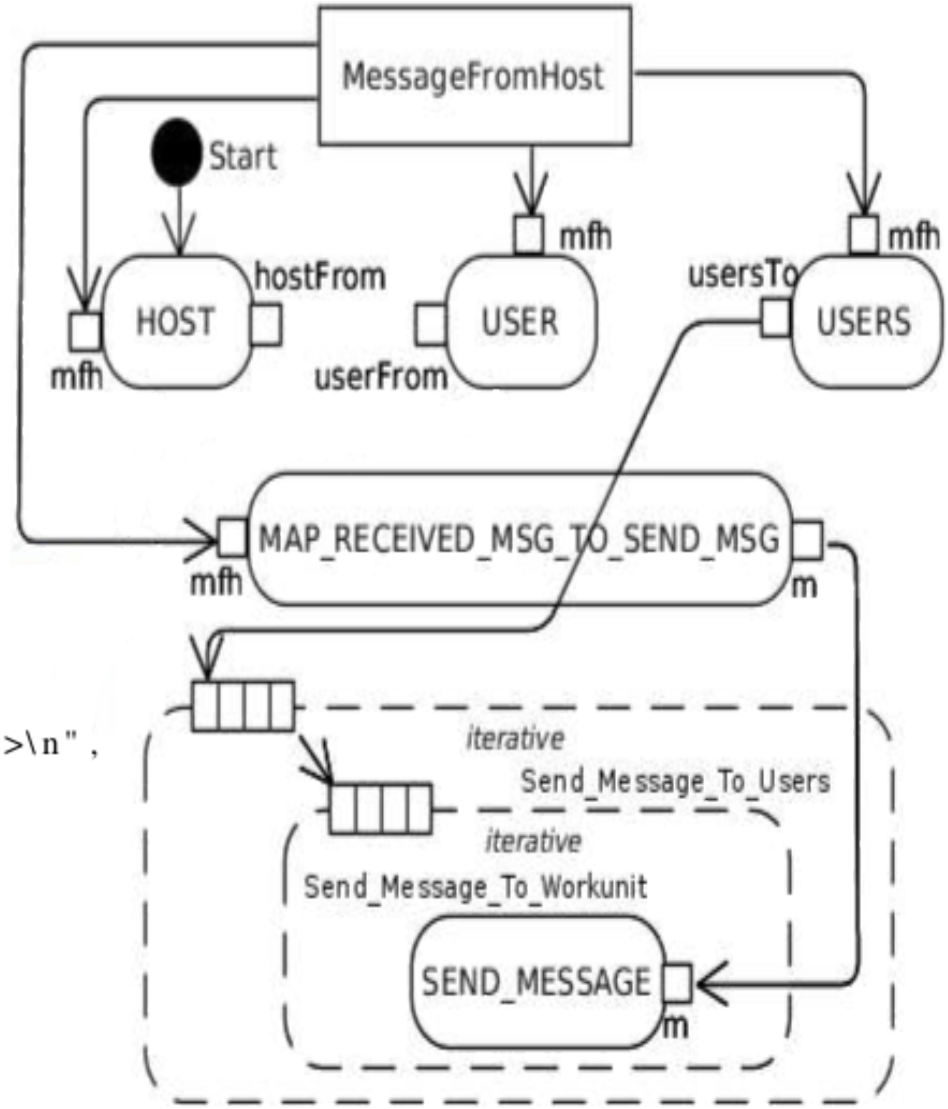


Figure 12. Code-mapping for UML4BOINC's «TrickleUp».

```
std::vector<std::string> messageFields;
/* ...fill messageFields with data! */
if(messageFields.size() >= 3) {
 char msgUp[1024] = {'\0'};
 snprintf(msgUp, 1024,
  "<to>%s</to>\n<from>%s</from>\n<msg>%s</msg>\n",
  messageFields[0],
  messageFields[1],
  messageFields[2]
}
int ret = boinc_send_trickle_up(
                (char*)variety, msgUp);
if(ret) { /* Error-handling, ... */ }
```

Listing 9. Trickle-message handling on the client-side.

Server-side



Code Generation Approaches

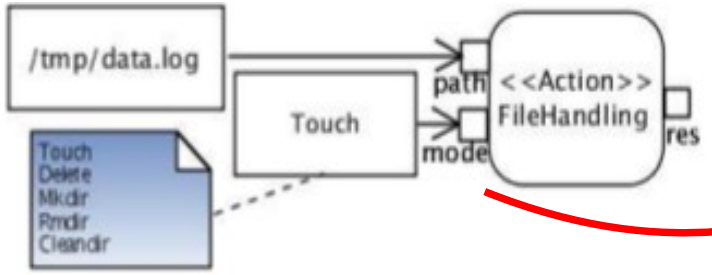


Figure 10. «Action» for file handling with two parameters.

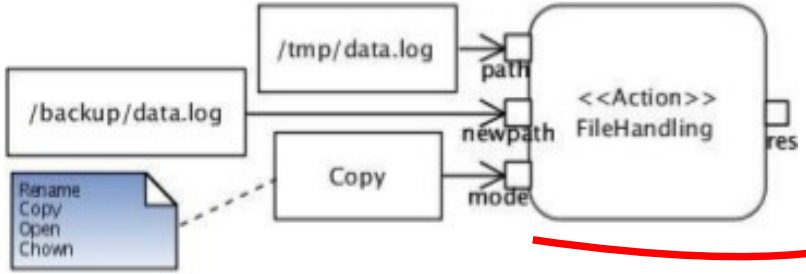


Figure 11. «Action» for file handling with three parameters.

Mode	BOINC's Function Call
one additional input pin	
Touch	int res = boinc_touch_file (path);
Delete	int res = boinc_delete_file (path);
Mkdir	int res = boinc_mkdir(path);
Rmdir	int res = boinc_rmdir(path);
Cleandir	int res = clean_out_dir (path);
two additional input pins	
Copy	int res = boinc_copy(path, newpath);
Chown	int res = boinc_chown(path, owner);
Rename	int res = boinc_rename(path, newpath);
Open	FILE *res = boinc_fopen(path, filemode);

Table II
MAPPING OF UML TO BOINC'S API.

- Easier recognition of functionality
- UML stereotypes can be replaced by specific icons (i.e. similar to Matlab or Simulink).

Code Generation Approaches

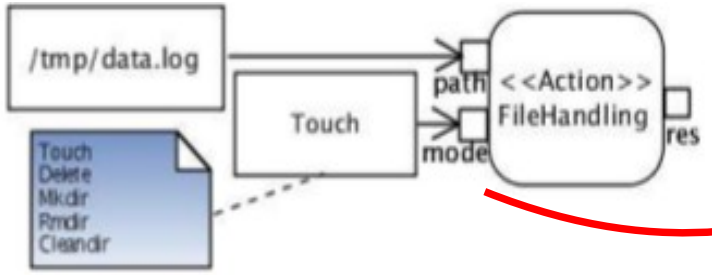


Figure 10. «Action» for file handling with two parameters.

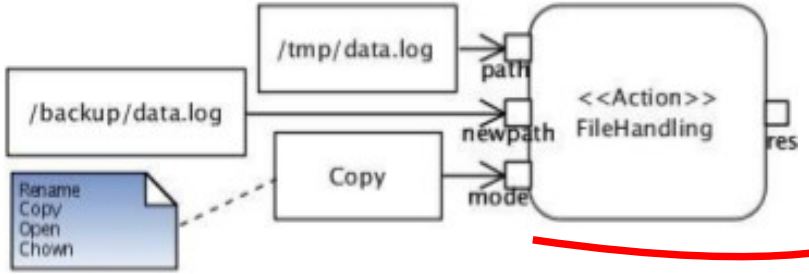


Figure 11. «Action» for file handling with three parameters.

Mode	BOINC's Function Call
one additional input pin	
Touch	int res = boinc_touch_file (path);
Delete	int res = boinc_delete_file (path);
Mkdir	int res = boinc_mkdir(path);
Rmdir	int res = boinc_rmdir(path);
Cleandir	int res = clean_out_dir (path);
two additional input pins	
Copy	int res = boinc_copy(path, newpath);
Chown	int res = boinc_chown(path, owner);
Rename	int res = boinc_rename(path, newpath);
Open	FILE *res = boinc_fopen(path, filemode);

Table II
MAPPING OF UML TO BOINC'S API.

- Easier recognition of functionality
- UML stereotypes can be replaced by specific icons (i.e. similar to Matlab or Simulink).

Thank you,

Christian Benjamin Ries

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