Batch execution of CAPRI tasks

- by Wolfgang Britz, July 2008 -

Why batch execution?

Currently, CAPRI tasks are typically defined and executed via the CAPRI GUI. The GUI supports the user by giving feedback to erroneous selection and reduces the selection space by excluding illegal settings. However, the GUI let the users to wait for execution of the current task before allowing starting a new one, and requires a environment supporting graphical user interaction. There may be instances where these restrictions need to be overcome:

- Repeatedly execution of a longer list of predefined tasks, such as a test suite to check the status of a certain work copy or the status of the trunk or a tag. The test may need to run a remote machine, and not during office hours. It may be started automatically if the trunk or a tag was updated during the working day. The batch execution ensures that all tests in the suite are executed, and nothing is left out.
- Execution of several scenarios in a row, e.g. for systematic sensitive analysis.
- Usage of an environment without graphical facilities, as in grid computing.

There had been also work around to circumvent the GUI, e.g. storing the GAMS code generated by the GUI under different names, and use OS command batch files to copy them, start GAMS, and copy the results again. However, for a larger test suite, such a proceeding is rather cumbersome. Further on, the batch execution facility provides also a testing environment for a more modular implementation of the CAPRI objects in Java.

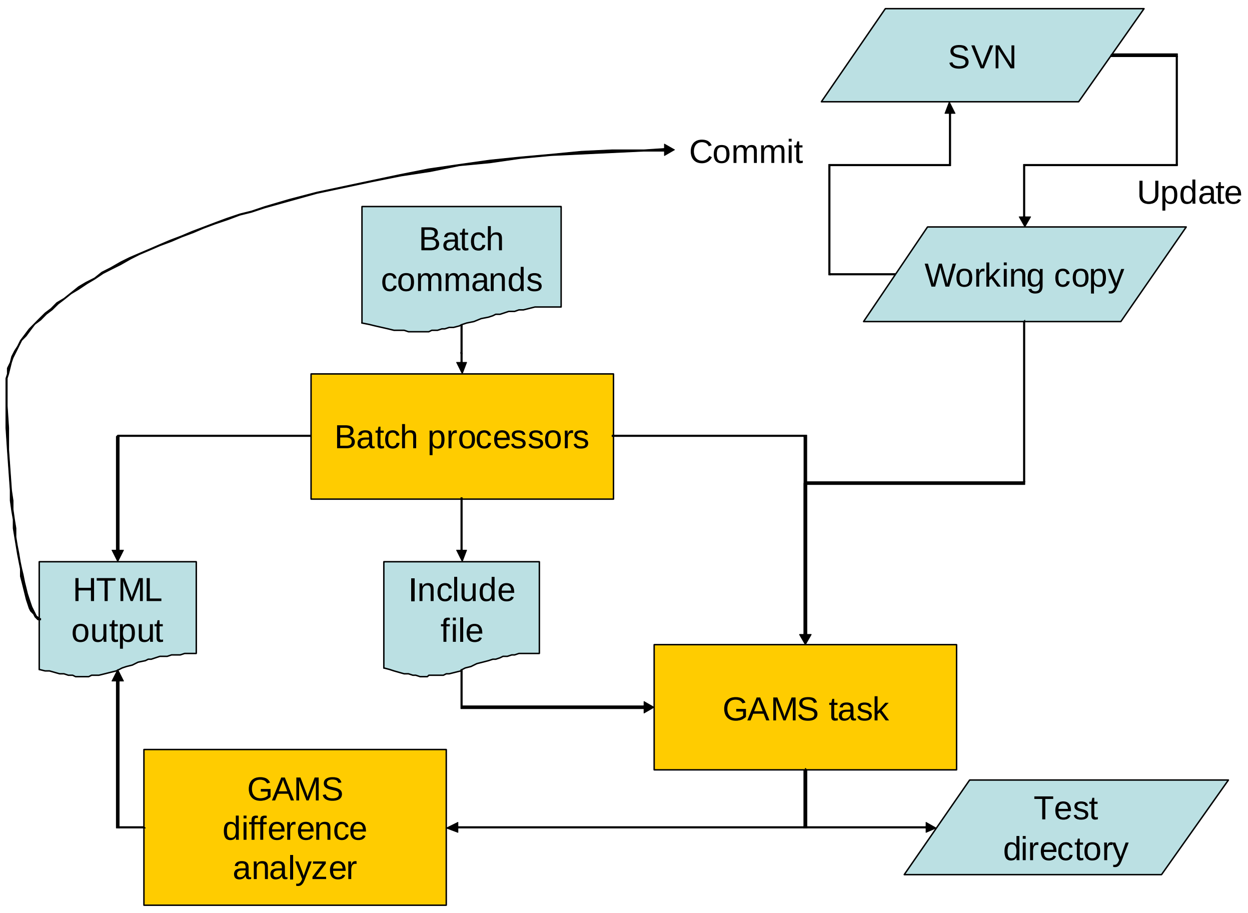
Overview on the batch processor and testing

The following diagram depicts the principal interaction in batch processing for testing purposes. Assume that the local working copy comprises some modifications which require testing. In order to do so, typically the result directories are first updated to be synchronized with the server to ensure a common comparison point.

Next, a suitable set of batch commands is edited in a text file, and the batch processor is started. It will generate from the commands in the text file objects with the required properties. These objects then output their settings (base year, Member states includes, model switches etc.) into an include file formatted according to the requirement of the GAMS task to be performed.

In opposite to normal execution mode, the GAMS task will store all output into a separate test directory. On error free completion, it will execute a GAMS based difference analyzer which compares the results from the working copy with the results in the test directory. The outcome of the test along with information about the run is reported to a HTML page and/or GDX files. After an inspection of the reports, the user will then take the decision to commit the changes or not.

Diagram: Conception overview of batch processing for testing



The batch command language

The batch handler reads the settings from a simple ASCII file. Each line is either a comment starting with an asterisk, or a setting line. A setting line comprises a key word, an equal sign, and the actual setting. The only exemptions are *ontext*, *offtext*, *exit* and comment lines starting with an asterisk. The following table lists the currently implemented key words. Key words may be concentrated or not, and any mixed of lower and upper case may be used.

Key word/phrase	**Meaning**
Base year	Defines the base year for the task, if applicable. Otherwise ignored
Execute	Will try to generate a task from the current settings, and excute it. Allowed settings: gamscompile or gamsexecute.
Exit	Will exit the current batch file
First year	Defines the first year for the task, if applicable. Otherwise ignored
Gams engine	Path to GAMS.exe (e.g. c:\programme\gams22.5\gams.exe)
Gams options	Gams options to use (e.g. RF=test.ref). KILL will remove any options present.
Last year	Defines the last year for the task, if applicable. Otherwise ignored
Member states	The list of member states, comma delimited. Further on, the following short cuts are allowed: all, EU27, EU25, EU 15, EU12, EU10, BUR, WBA
Model switches	MARKET_M, YANI_M, RECDYN; further switches as BASELINE, POLSHIFT, EXPOST are generated from the task. KILL removes all model switches currently present
Number of iterations	Maximal number of iterations between supply and market parts
Number of processors	Maximal number of processors to use when applying the grid solution feature of GAMS
Offtext	Statements are interpreted again
Ontext	Any following statements are ignored
Output dir	The directory where listing files will be generated along with the HTML summary page

Key word/phrase	**Meaning**
Regional break down	Member States, NUTS 2, Farm types. Not all settings may be allowed depending on the task
Res dir	Results where results will be read from and stored to
restartOutDir	Substitute for restart directory
resultsOutDir	Substitute for results directory
Scen description	A string describing the scenario
Scen name	The name of the scenario, determines at the same time the policy file to use
Scr dir	Scratch directory used by GAMS and by the task to store temporary files.
Sim year	The simulation year
Task	The task name
User	Name of the user, will be added to the meta data for the run
Work dir	The root of the CAPRI implementation from where the main program will be started, and how includes are resolved.
Work step	The work step for the task

Remark:

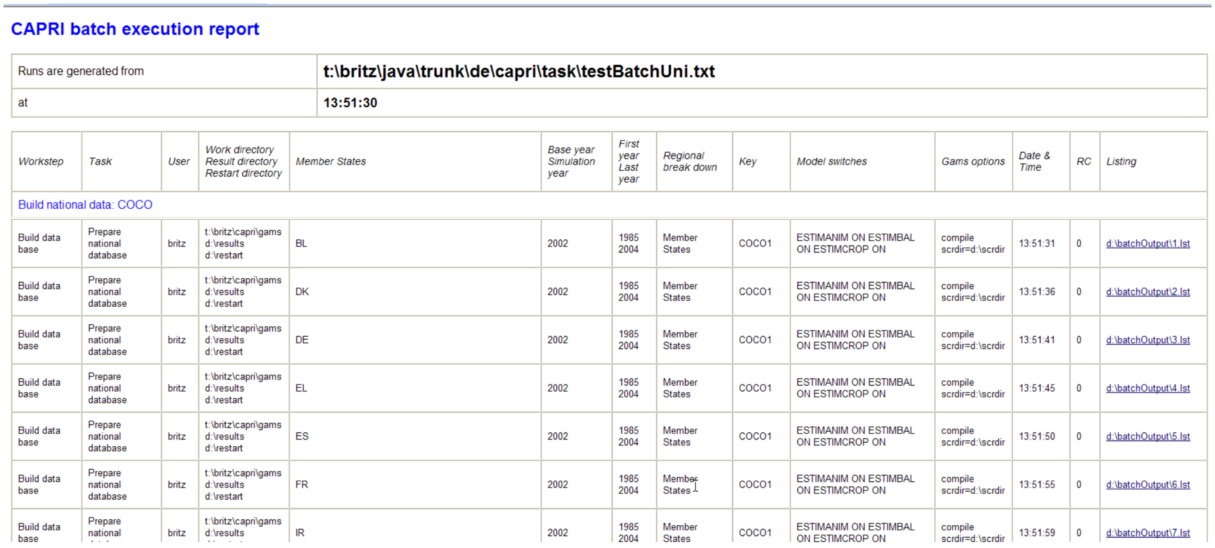
- Neither key words nor settings are case sensitive. Key phrases can be written as a single word, also.
- Comment blocking surrounding by “*-----“ are reported to the HTML output page.
- Any error while processing the batch file is reported to the HTML output page.
- Execution after errors will continue.

- Errors while defining a task will prevent it from executing. The same holds from using illegal GAMS options, as they will trigger an error even preventing compilation.

The output from batch execution

As it is assumed that batch execution will not be monitored by the user during execution, some logging mechanism must be established. The current implementation offers two interlinked approaches:

1. The listing files generated by GAMS, and the include files steering the GAMS programs as “fortran.gms” are stored under a specific id in the current work directory, or an directory defined by the user by the key word “copy dir”.
2. A HTML page reports all tasks which have been started, the return code of the GAMS process and all major setting, as well as link to open the listing file with the editor. The following screen shot shows the first part of the HTML page resulting from executing the above batch file. Tasks which did yield a non-zero GAMS return code and errors are shown in red.



CAPRI batch execution report

Runs are generated from	**t:\britz\java\trunk\de\capri\task\testBatchUni.txt**
at	**13:51:30**

Workstep	*Task*	*User*	*Work directory Result directory Restart directory*	*Member States*	*Base year Simulation year*	*First year Last year*	*Regional break down*	*Key*	*Model switches*	*Gams options*	*Date & Time*	*RC*	*Listing*
Build national data: COCO													
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	BL	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:31	0	d:\batchOutput\1.lst
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	DK	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:36	0	d:\batchOutput\2.lst
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	DE	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:41	0	d:\batchOutput\3.lst
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	EL	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:45	0	d:\batchOutput\4.lst
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	ES	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:50	0	d:\batchOutput\5.lst
Build data base	Prepare national database	britz	t:\britz\capri\gams d:\results d:\restart	FR	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:55	0	d:\batchOutput\6.lst
Build data base	Prepare national	britz	t:\britz\capri\gams d:\results	IR	2002	1985 2004	Member States	COCO1	ESTIMANIM ON ESTIMBAL ON ESTIMCROP ON	compile scrdir=d:\scrdir	13:51:59	0	d:\batchOutput\7.lst

The current batch steering file used for test

```
*********************************************************************
*
* Standard test suite for the GAMS programs of CAPRI
*
* Since: July 2008
* Author: Wolfgang Britz
*
*********************************************************************

 base Year =  2002
 first Year = 1985
 last Year  = 2004

 member States=all
 regional Break Down = NUTS 2
 gams engine=c:\programme\gams22.8\gams.exe
 user = britz

 work dir = t:\britz\capri\gams
*
* --- where the HTML page and the listings
*     will be stored
*
 output dir = d:\batchOutput

 res dir = t:\britz\capri\results
 scr dir = d:\scrdir

 gams options = scrdir=d:\scrdir
*
* --- The following settings will write
*     The results into different directories.
*     The directory structure will be automatically
*     generated
*

 restartOutDir = d:\restart
 resultsOutDir = d:\results

 number of processors = 4
 number of iterations = 1

*---------------------------------------------------------
*
* Build national data: COCO
*
*---------------------------------------------------------

 model switch=ESTIMCROP ON
 model switch=ESTIMANIM ON
 model switch=ESTIMBAL ON

 task= Prepare  national database
 execute=gamscompile

 task= Finish  national database
 execute=gamscompile

 model switch=kill

*---------------------------------------------------------
*
* Build global data: GLOBAL
*
*---------------------------------------------------------


 task= Build global database
 execute=gamscompile

*---------------------------------------------------------
*
* Build regional data base : CAPREG
```

```
*
*----------------------------------------------------------


 task= Build regional database
 execute=gamscompile


*----------------------------------------------------------
*
* Work steps of build data: CAPDIS
*
*----------------------------------------------------------

 task= Build HSMU database
 execute=gamscompile

*----------------------------------------------------------
*
* Baseline generation
*
*----------------------------------------------------------

 last Year  = 2013
 sim Year = 2013
 scen name=MTRSTD
 model switch=MARKET_M ON
 model switch=YANI_M ON

 regional Break Down = NUTS 2

 number of iterations = 15
 task= Generate expost results
 execute=gamscompile

 task= Generate policy shifts
 execute=gamscompile

 task= Generate trend projection
 execute=gamscompile

 regional Break Down = Member States

 number of iterations = 1
 model switch=YANI_M OFF
 task= Baseline calibration
 execute=gamscompile

 model switch=MARKET_M OFF
 regional Break Down = NUTS 2

 task= Baseline calibration
 execute=gamscompile

*----------------------------------------------------------
*
* Simulation
*
*----------------------------------------------------------

 model switch=MARKET_M ON
 model switch=YANI_M   ON
 model switch=RECDYN  OFF

 regional Break Down = NUTS 2

 task= Run simulation
 execute=gamscompile

 scen name=AGENDA
 task= Run simulation
 execute=gamscompile

 scen name=WTOHRB
 task= Run simulation
 execute=gamscompile
```

The file above will in total trigger 112 compilation tests, starting all programs which can be accessed via the GUI. Alternatively, the batch could be used to generate all production data from scratch.

Towards a test suite for CAPRI

Some issues are still open to realize a test suite for CAPRI. Straightforward are compile checks – but even those are currently not systematically undertaken after commits to the trunk. As compilation does not overwrite any production data, they can be simply performed on the current working copy, or an updated version synchronised with the trunk. However, we are still missing a batch update of an SVN installation, albeit it is by now rather straightforward since the SVNKit has already been successfully tested for documentation tool.

Running the programs in order to detect probably run-time errors raised by GAMS carries already the risk to overwrite production data. Therefore, the "resultsOutDir" and "restartOutDir" options are introduced. The GAMS code is modified so that write statements to "RESDIR" are replaced to output to the "resultsOutDir", whereas input is taken from RESDIR. Accordingly, all existing results will be preserved.

The most tricky issue is certainly to judge if differences in numerical results between implementation are sizeable or not, and if they are, if one of the implementation is superior. The latter will in the near future certainly not be answered based on an algorithm. But it may be possible to develop for each task a small GAMS program which analyses and reports differences in major results.

Basically, we need for GLOBAL, COCO, CAPREG, CAPTRD and CAPMOD a possible included program, which loads the very same results set resulting from the current task and settings for the current run and a comparison one (typically the one in local working copy, which may be synchronised with the trunk). In practise, that could be the result directory for output (current run) and input (previous results). That features is implemented now in the batch processor, and in the GAMS code, but more thorough testing is necessary, and eventually, also some discussions regarding the detailed implementation.

The program would calculate absolute or relative differences between the two sets for a predefined array of positions, and report those, either as a GDX or as a HTML page. The positions selected should be good indicators in the sense that they (a) allow to judge quickly if the results has changed at all (quite important if refactoring is done to speed up processing or introduce clearer structure), and (b) detect major changes.

Technical implementation

The implementation is based on Java, and actually was realized not so much as a batch execution device. Rather it started as a testing environment for a new IT structure for the CAPRI GUI. The idea behind the new implementation is a clearer separation of the logical sphere of CAPRI – its business objects – and the man-machine interface which defines and execute tasks. The current GUI has emerged from some Java tests when the CAPRI was still hosted in a FORTRAN/C-GAMS environment, and has grown considerably without ever being redesigned. The development of the batch processing abilities is a first step to a more modular and encapsulated implementation of CAPRI tasks. Clearly, that will ease in future alternative implementations, e.g. in a client-server environment or an alternative processing environment

At the core are executable object of type *AgpTask*. Each AgpTask has properties (task as “Build regional database”, a work step “Build data base”, the GAMS program to execute (CAPREG), and further properties as “base year”, “first year”, “last year”, the list of Member States etc). An *AgpTaskHandler* is an interface which is able to generate and execute such a task. Currently, there is abstract implementation *AgpDefaultTaskHandler* which implements major elements of the interface, from which a batch execution handler is derived. But, in the near future, the CAPRI GUI will be replaced by a *AgpTaskHandler* implementation.

The execution of a CAPRI tasks is currently realised as a GAMS process. In future, also the exploitation steps will be realized.

Outstanding issues

- Timed execution
- Automated update and revert of the local working copy

- Distribution and start on remote machine