



GMO: GAMS' Next-Generation Model API

Steve Dirkse

sdirkse@gams.com

GAMS Development Corporation

www.gams.com





GMO: A Team Effort

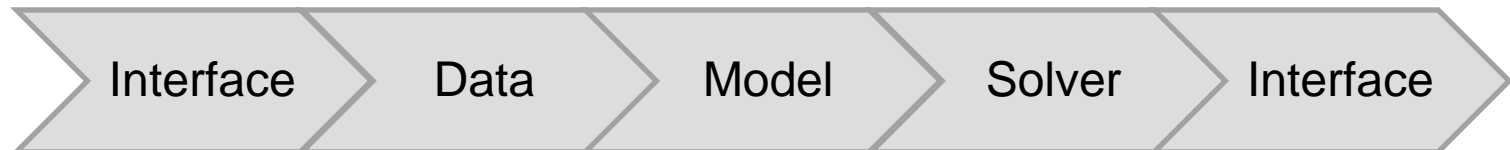
- Michael Bussieck
 - Jan-Hendrik Jagla
 - Alex Meeraus
 - Paul van der Eijk
 - Lutz Westermann
-
- Team projects introduce challenges and advantages
→ more on this later



Background

GAMS' Fundamental concepts

- Different layers with separation of
 - model and data
 - model and solution methods
 - model and operating system
 - model and interface





Background

GAMS' Fundamental concepts

- Different layers with separation of
 - model and data
 - model and solution methods
 - model and operating system
 - model and interface





Solver Links – Different Perspectives

GAMS User

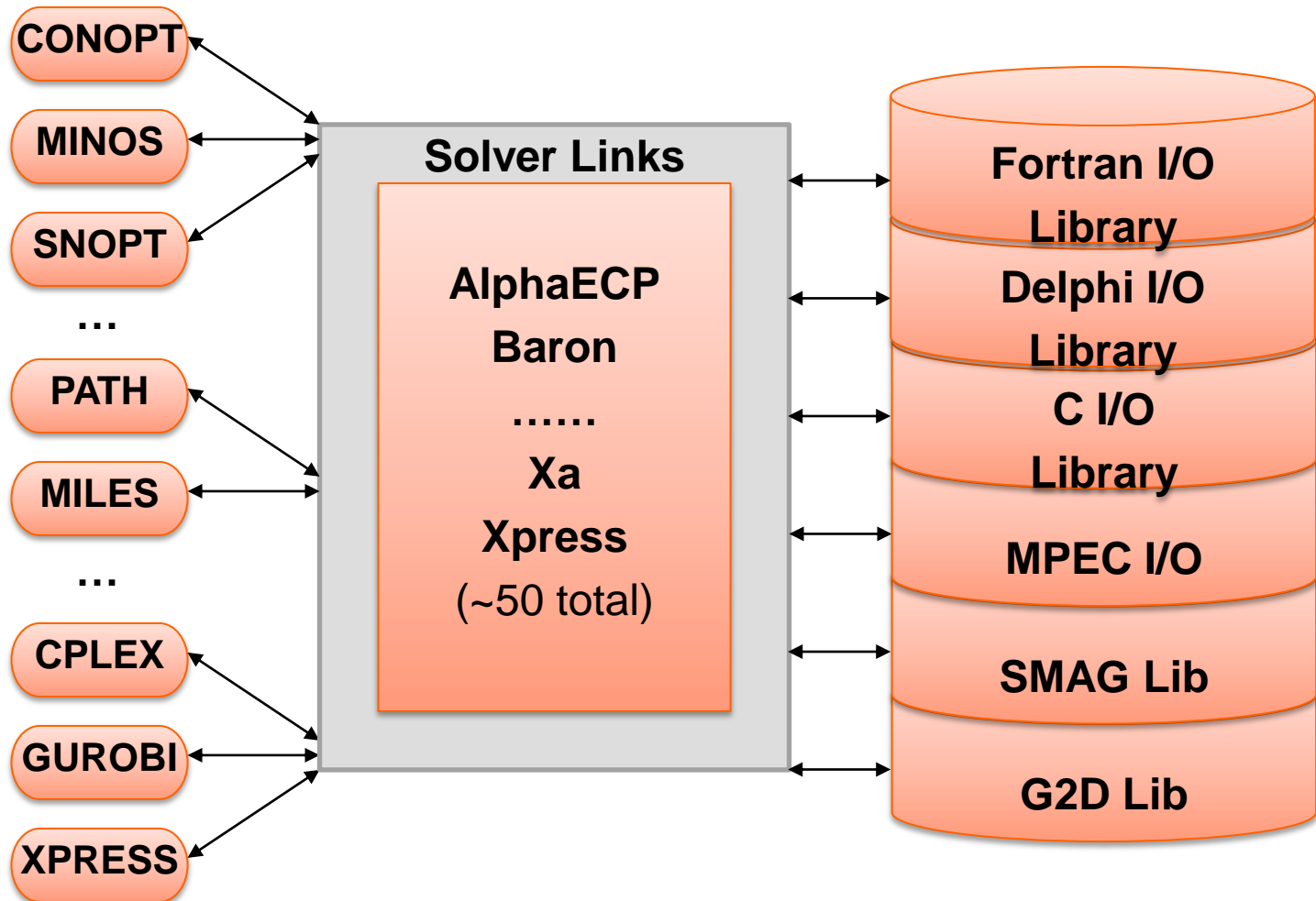
- Standardized solver interface allows “hassle free” replacement of solvers: *option nlp=conopt;*

Solver & Solver-link Developer – *our focus here*

- IO Library provides access to
 - Matrix
 - Function/Gradient/Hessian evaluations
 - Solution file writer
 - Output handling
 - GAMS Options (e.g. resource limit)
 - Problem attributes (SOS, semicont, semiint, priorities, scales)
 - Utility routines - problem rewriting, matrix reordering, SBB, ...



Reuse? What's that?!?





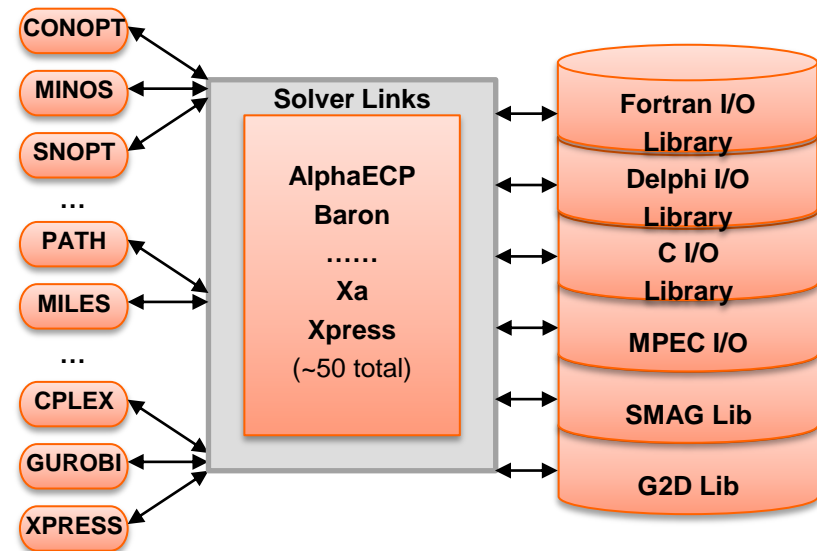
Talk Outline

- Why a new model API?
- What do we need it to do?
- What does it look like? How is it put together?
- How did we do it?
- When are we going to be finished?



Multiple I/O Libraries - Advantages

- proven over many years
- all platforms supported
- all GAMS-features available
- written by language experts, use all language features
- resulted in high-quality links across solvers and platforms
→ has been one factor in our success

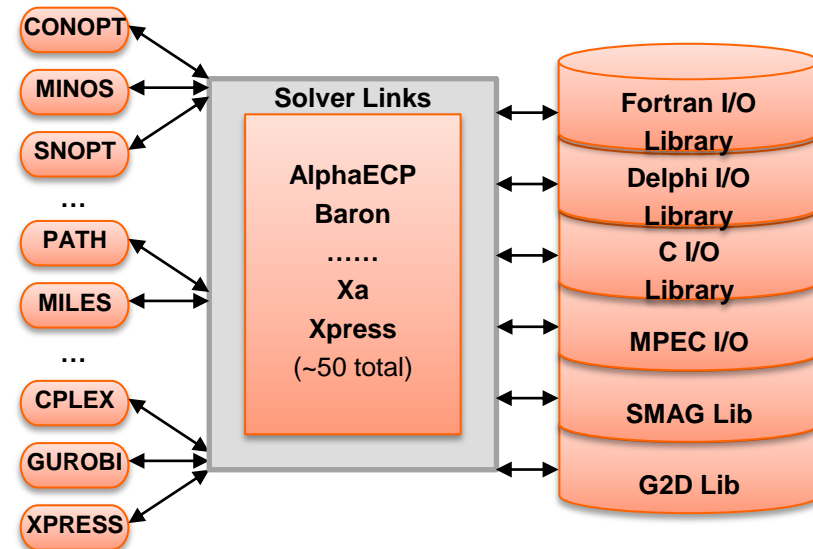




Multiple I/O Libraries - Disadvantages

- Not always intuitive to use
Linking your Solver to GAMS
- THE COMPLETE NOTES
(160 pages !!)

- Outdated design – I/O, STOP
- feature-poor (e.g. no automatic reformulation of objective func/var)
- inconvenient & expensive to maintain
- painful to move ‘inert mass’ forward – **not agile**
- linking your solver (without buddy at GAMS) is very difficult





Philosophy behind GMO

- Then
 - Computing environment: limited time and memory
 - Algorithm APIs not uniform or language-neutral
 - Expert users who understand optimization
 - *Don't use unnecessary space or time*
 - *If the link gets in trouble, just abort, the user will fix things up and re-run.*
- Now
 - Most users won't hit space/time limits
 - APIs look similar, are language-neutral
 - Users may be domain experts, not MP experts
 - *Use of additional space & time to give the GMO and GAMS user a better experience is justified.*



Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement **once**, run **everywhere** (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models



Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models



GMO: Powerful and Convenient API

- What's a powerful call?
 - Basic CS: information hiding, encapsulation, object model, abstraction
 - One call to do the job required, e.g. Hessian setup
 - No preconditions, magic calls, or nasty side effects
 - Should handle failure gracefully
- Convenience - multiple routines and “flavors”
 - Jacobian row- vs. column-wise, tuples
 - Objective reformulation – function or variable
 - Free rows - yes or no
 - Column evals: dense or sparse, all or just NL
 - Common/typical tasks done in GMO, not the link



Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models



Solver Integration

```
solve mymodel minimizing z using lp  
mymodel.solveLink = {ChainScript, CallScript,  
    CallModule, AsyncGrid, AsyncSimulate, LoadLibrary};
```

- ChainScript: Solver process, GAMS vacates memory
 - + Maximum memory available to solver
 - + protection against solver failure (*hostile* link)
 - swap to disk
- Call{Script/Module}: Solver process, GAMS stays live
 - + protection against solver failure (*hostile* link)
 - + no swap of GAMS database
 - file based model communication



Solver Integration – cont.

- LoadLibrary: Solver DLL in GAMS process
 - + fast memory based model communication
 - + update of model object inside the solver (hot start)
 - not (yet) supported by all solvers
- trnsport.gms (LP) solved 500 times with CPLEX:

```
set ss /s1*s500/;    loop {ss,  
    solve transport minimizing z using lp};
```

- | | |
|----------------|------------------|
| – ChainScript: | 33.04 s (28.9s)* |
| – CallModule: | 13.78 s (12.7s) |
| – LoadLibrary: | 2.37 s (2.0s) |
| – Hot Start: | 0.37 s (0.4s) |

Cplex simplex
time: 0.2 s

* without Virus Scanner



Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement **once**, run **everywhere** (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models



Implement Once, Run Everywhere

- All GMO coding done in a single language and style
 - Allows code sharing with other components
 - Allows for shared development
- All GMO coding is platform-independent
 - Makes writing code faster, more reliable
 - Maintenance is simplified
- Platform-dependent code isolated in utility libraries
 - Makes adding a new platform easier
 - Maintenance is simplified
 - Unit testing is easy and effective



Automated Generation of APIs

‘The GAMS Wrapper’

- API is defined using the GAMS language
- A tool written in GAMS is used to regenerate APIs for all languages
- Executed on request and nightly

```

* Properties reading/writing data that originally came from control vectors
gmoModelType .int. (r.getModelType ,v.modeltype )
gmoM .int. r .GetRows
gmoN .int. r .GetCols
gmoScaleOpt .int. ((r,w).ScaleOpt )
gmoSense .int. (r.direction ,w.setObjSense )
gmoObjVar .int. (r .GetObjVar ,w.setObjVar )
gmoOptFile .int. ((r,w).OptFile )
gmoPriorOpt .int. ((r,w).prioropt )
gmoNLConst .int. ((r,w).nlconst )
gmoNZ .int. (r .GetNonZeros,w.NZ )
gmoNLNZ .int. r .GetNLNonZeros
gmoNLM .int. r .GetNLRows
gmoNLC .int. r .GetNLCols
gmoObjRow .int. r .GetObjRow
gmoDictionary .int. ((r,w).Dictionary )
gmoHaveBasis .int. ((r,w).havebasis )

gmoNameOptFile .oSS. ( r .NameOptFile ,w.SetNameOptFile )
gmoNameSolFile .oSS. ( r .NameSolFile ,w.SetNameSolFile )
gmoNameZLib .oSS. ( r .NameZLib ,w.SetNameZLib )
gmoNameMatFile .oSS. ( r .NameMatFile ,w.SetNameMatFile )
gmoNameDict .oSS. ( r .NameDict ,w.SetNameDict )
gmoNameParams .oSS. ( r .NameParams ,w.SetNameParams )
gmoNameInput .oSS. ( r .NameInput )

/
set f(en,tp,es,ta) function and procedures /
gmoLoadDataLegacy .(0,result.int,1,msg.oSS)
gmoInitData .(0,result.int,1,rows.int, 2,cols.int)
gmoCompleteData .(0,result.int,1,instance.CSS)
gmoQMaker .(0,result.int,1,density.B)
gmoGetObjQ .(0,result.int,1,colIdx.PLIA,2,rowIdx.PLIA,3,coef.PDA)
  
```

- A change in the definition of the API immediately makes it into all language interfaces
- No manual and therefore error-prone efforts required



Automated Generation of APIs

‘The GAMS Wrapper’

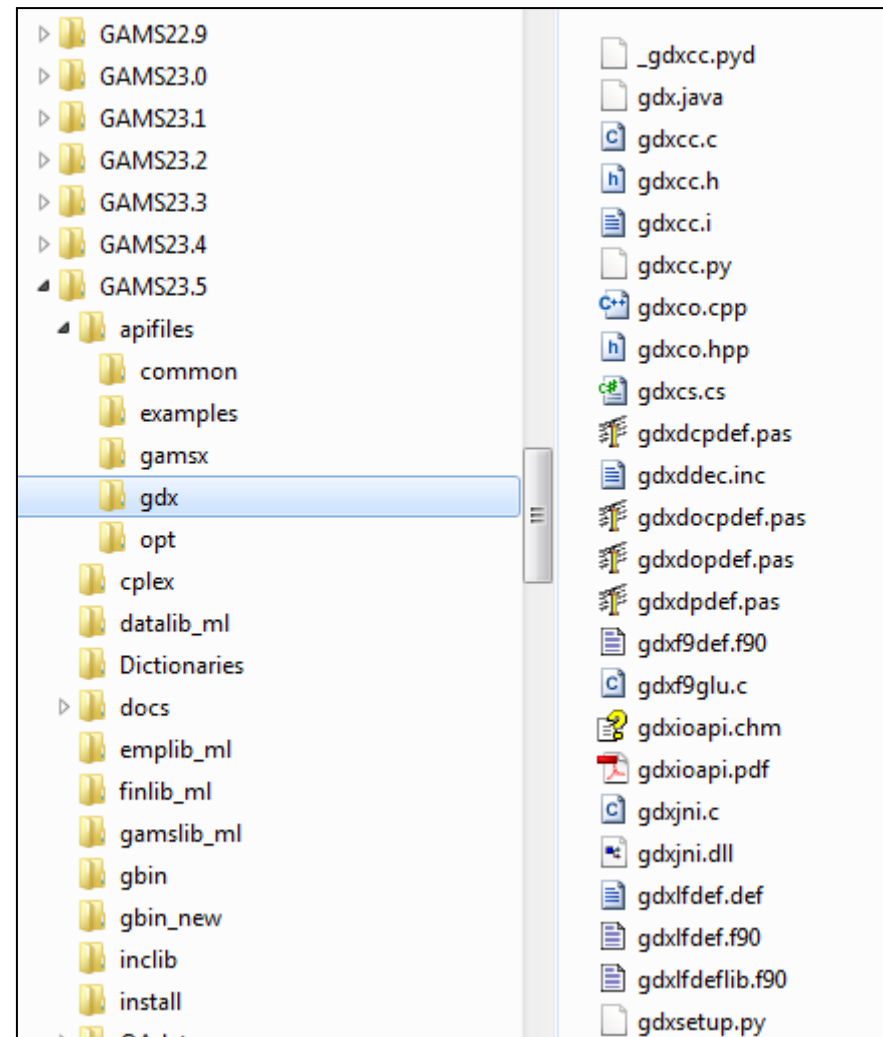
- Automated nightly testing
- API version checks
- Reusable for multiple GAMS component libraries
 - GMO
 - GAMS
 - GDX
 - Option

The screenshot shows a window titled 'gmside: C:\tmp\gpr [C:\home\jan\8_alpha\src\laphwrap\gmoapi.gms]'. The main area displays the code for 'gmoapi.gms'. The code is organized into two columns. The left column lists various GAMS model attributes and their corresponding GAMS commands, such as 'gmoModelType', 'gmoM', 'gmoN', 'gmoScaleOpt', 'gmoSense', 'gmoObjVar', 'gmoOptFile', 'gmoPriorOpt', 'gmoNLConst', 'gmoNZ', 'gmoNLNZ', 'gmoNLM', 'gmoNLN', 'gmoObjRow', 'gmoDictionary', 'gmoHaveBasis', 'gmoNameOptFile', 'gmoNameSolFile', 'gmoNameZLib', 'gmoNameMatFile', 'gmoNameDict', 'gmoNameParam', and 'gmoNameInput'. The right column lists the corresponding GAMS commands, such as 'Model Type', 'Number of equations', 'Number of variables', 'Scaling Flag', 'Direction of optimisation', 'Objective variable index', 'Optfile Number', 'Priority Flag', 'length of NL constant pool', 'Number of non zeros in constraints', 'Number of nonlinear non zeros in constr', 'Number of nonlinear rows', 'Number of nonlinear columns', 'Objective row index', 'Dictionary file written', 'Do we have basis', 'Option file name', 'Solution file name', 'External Function Library Name', 'Matrix file name', 'Dictionary file name', 'Params file name', and 'Input file name'. The bottom of the window shows a status bar with '1:1' and 'Insert'.



Distributed GAMS APIs

- Component Libraries
 - GAMS
 - GDX
 - Option
- Supported languages
 - C, C++, C#
 - Delphi
 - Fortran
 - Java
 - VBA, VB.Net
 - Python
- Examples/Documentation





Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- **Support meta-solvers (e.g. DICOPT, SBB, Examiner)**
- **Separate Model from Environment**
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models



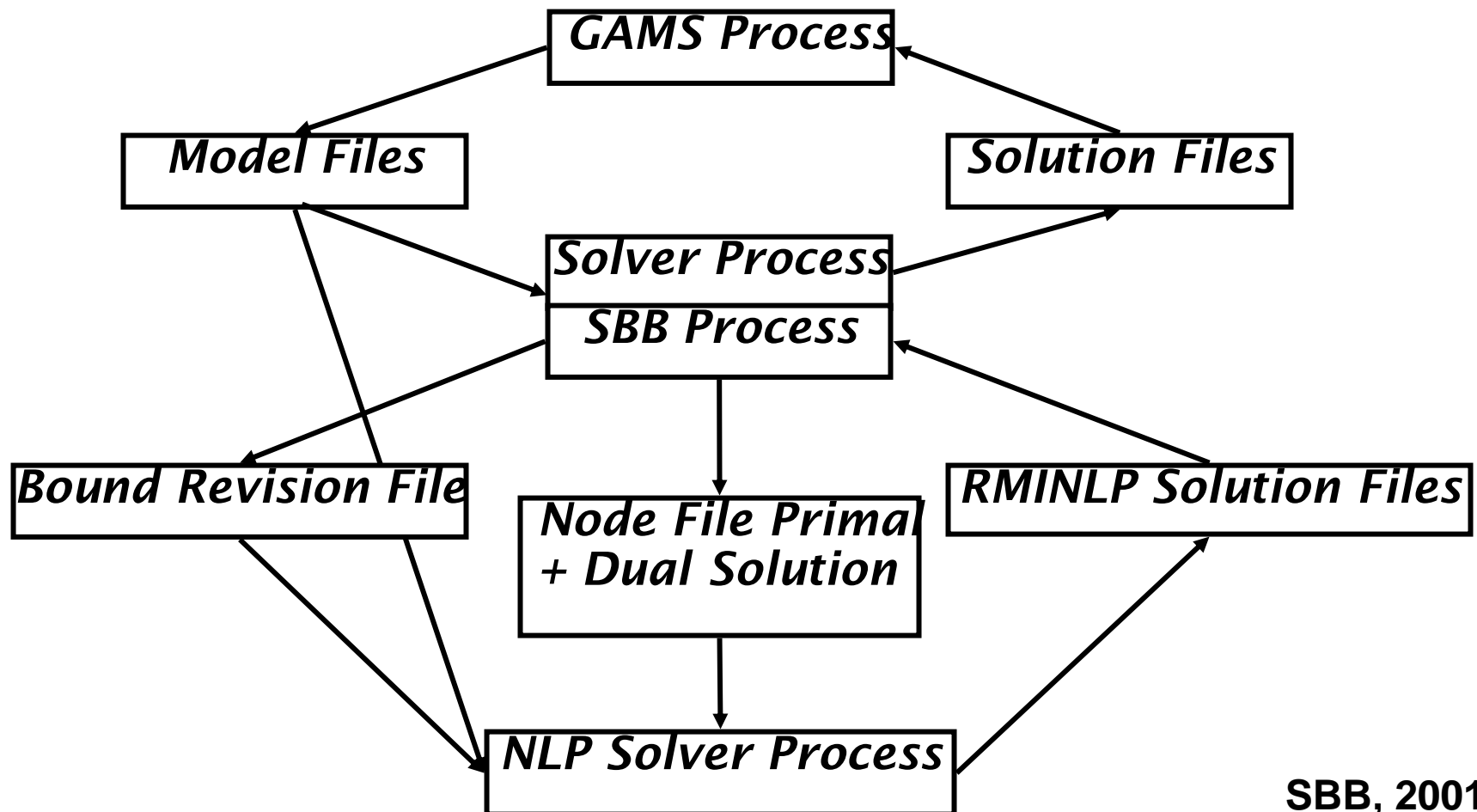
Separating Model & Environment

- Detach GMO from 'GAMS Environment'
- Ease linking of experimental solvers to GAMS
- Simplify use of GAMS as one piece of a puzzle

- E.g. sophisticated solvers use basic MP technology:
 - SBB (B&B requires NLP technology)
 - DICOPT (OA requires NLP+MIP)
 - BARON (requires NLP+LP)
 - LogMip (NLP+MIP)



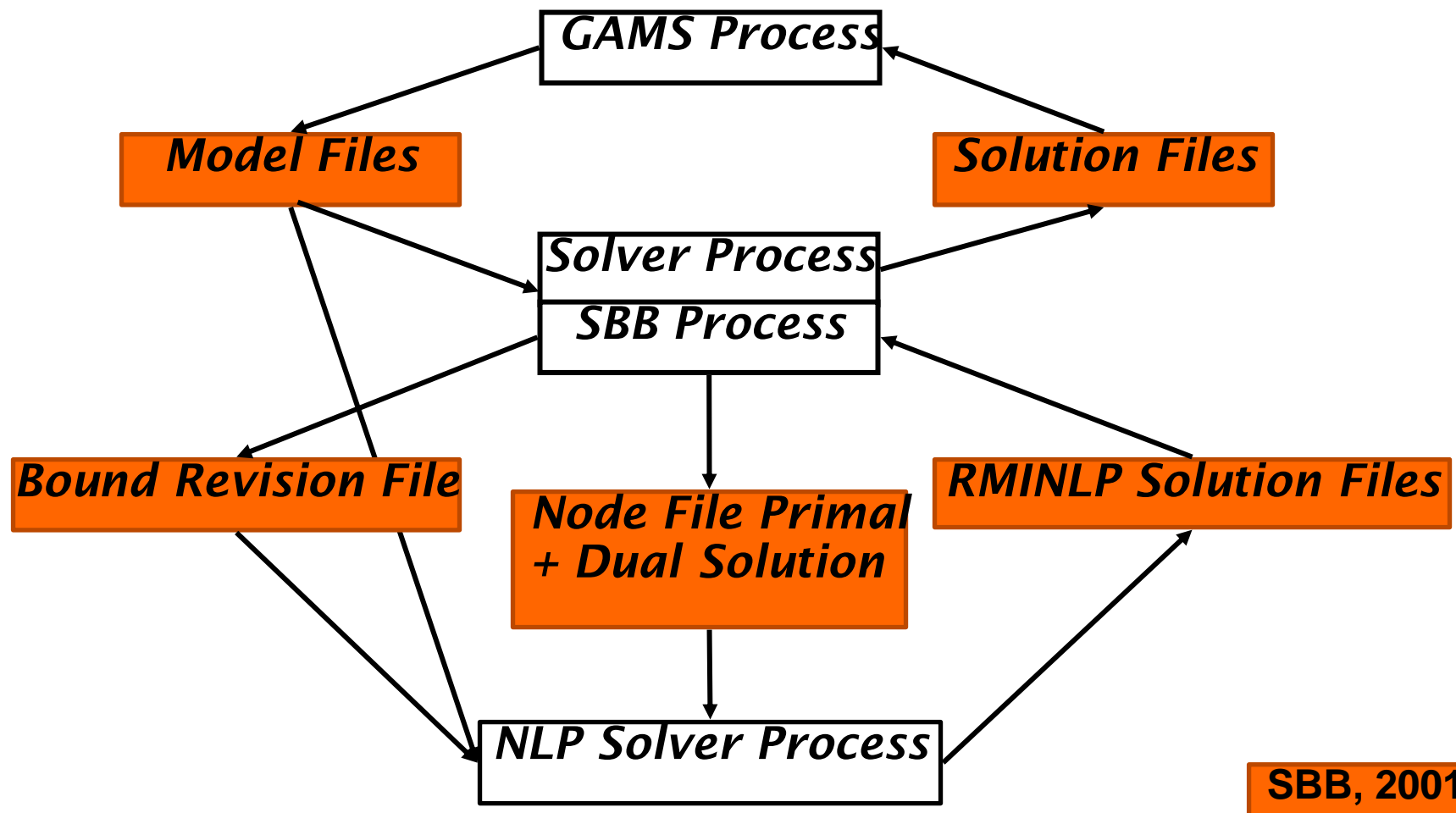
'Efficient' Implementation of B&B



SBB, 2001



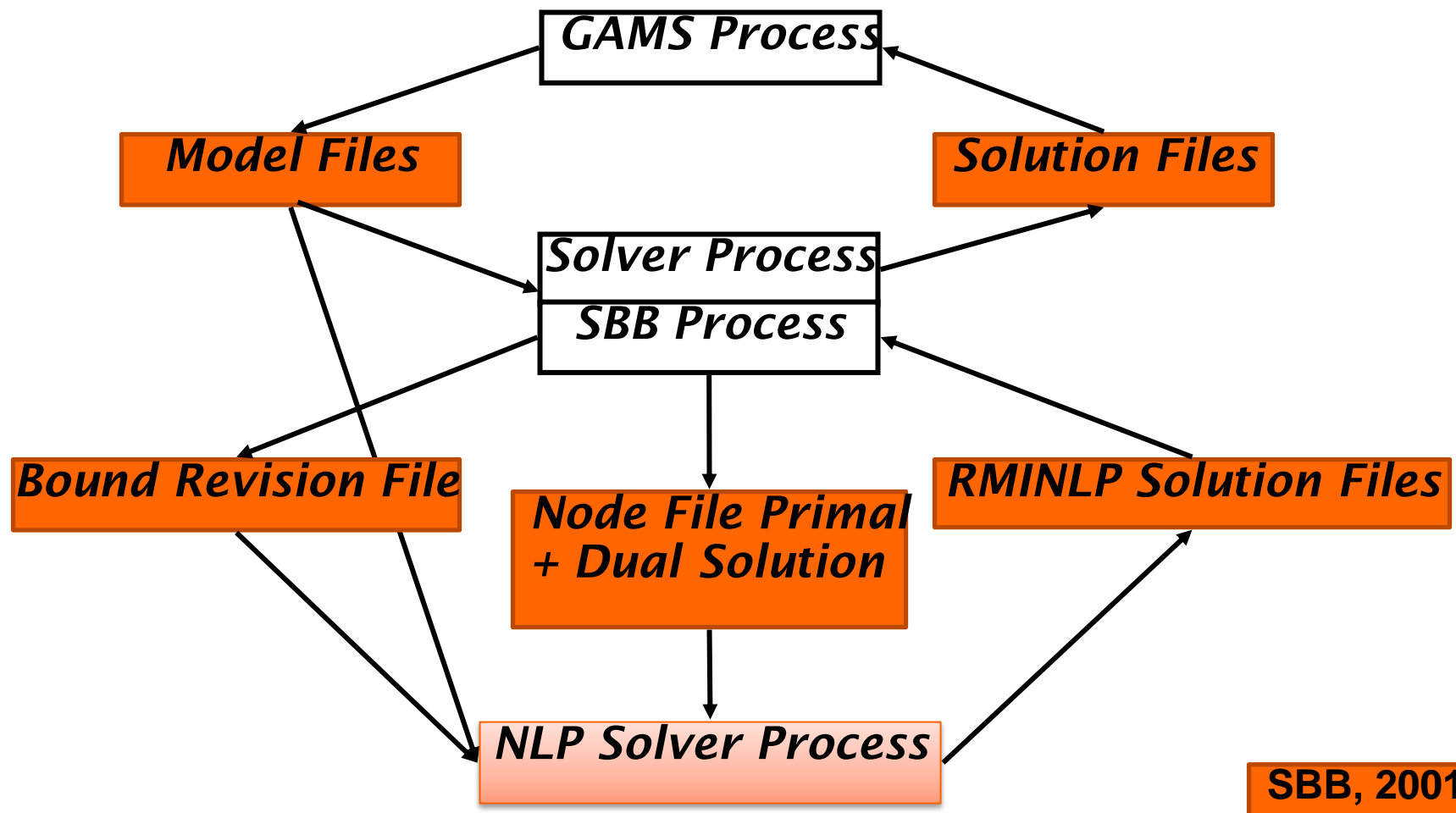
'Efficient' Implementation of B&B



SBB, 2001



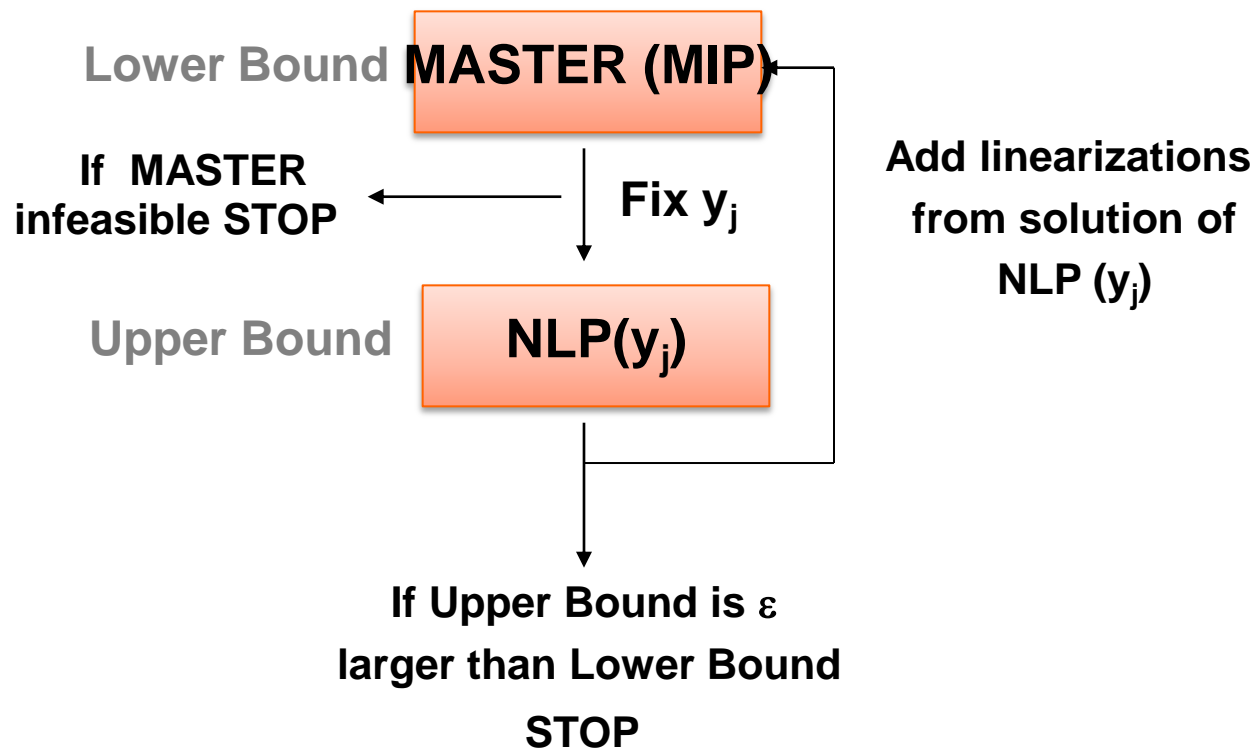
'Efficient' Implementation of B&B



SBB, 2001



Dicopt (Outer Approximation)





Series of NLP and MIP solves

```

--- DICOPT: Log File:
Major Major      Objective      CPU time      Itera-  Evaluation  Solver
Step  Iter      Function      (Sec)      tions      Errors
NLP    1          1.04923          0.02          38           0       conopt
MIP    1          9.07274          0.09          28           0       cplex
NLP    2        *Infeas*          0.00          10           0       conopt
MIP    2        13.02091          0.13          32           0       cplex
NLP    3          1.26864<          0.03          27           0       conopt
MIP    3        13.93760          0.11          29           0       cplex
NLP    4        *Infeas*          0.02           7           0       conopt
MIP    4        13.99258          0.11          19           0       cplex
NLP    5        *Infeas*          0.02          13           0       conopt
MIP    5        21.03812          0.11          23           0       cplex
NLP    6          1.26864          0.02          17           0       conopt
--- DICOPT: Terminating...
  
```

- Lots of file writing and reading to communicate between Dicopt, MIP, and NLP solver
- Basically start a whole new process over and over



New Dicopt Implementation

Joined work with Ignacio Grossmann, Juan Pablo Ruiz
(Carnegie Mellon University)

- Object Oriented
- Use C++ Interface to GMO
- Use standardized solver interface to call NLP/MIP solver in-core (pass GMO 'handle' on to solver)
- Algorithmic improvements



Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
 - Platform-independent code, isolate the “dirty bits”.
 - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- **Comprehensive – one-stop shop for all linking needs**
- **Support shared-library implementation of solver links**
- **Support multiple models**



How We Did It: SVN

Log Messages - C:\repos\source\products\trunk\src\gmoxxx

From: 8/13/2010 To: 11/ 4/2010 Messages, authors and paths

Revision	Actions	Author	Date	Message
20991		jhjagla	4:52:16 AM, Thursday, November 04, 2010	filter tiny marginals
20948		jhjagla	8:45:31 AM, Wednesday, November 03, 2010	fix
20947		jhjagla	8:27:52 AM, Wednesday, November 03, 2010	cleanup when to check for objfun and/or equ
20942		jhjagla	6:03:16 AM, Wednesday, November 03, 2010	never reset objval for short solve statement
20938		mbussieck	5:42:21 AM, Wednesday, November 03, 2010	treat objvar same as objrow (valnaint)
20934		jhjagla	4:47:23 AM, Wednesday, November 03, 2010	gmoDirtyGetObjFNLInstr needs objequ/fun
20933		jhjagla	4:43:12 AM, Wednesday, November 03, 2010	fix dumping of control file for mcp,cns, add gmovalnaint prop
20913		jhjagla	9:29:01 AM, Tuesday, November 02, 2010	even if the model is all linear, we might have to update the skips
20912		jhjagla	9:14:28 AM, Tuesday, November 02, 2010	write FATAL error messages also to stdout so we can catch it in the tests
20906		jhjagla	4:22:42 AM, Tuesday, November 02, 2010	allow for variable in constraint names, make use of variable id when using same

filter tiny marginals

Showing 100 revision(s), from revision 19311 to revision 20991 - 1 revision(s) selected.

☒ Hide unrelated changed paths

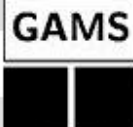
☐ Stop on copy/rename

☐ Include merged revisions

- Everybody agrees: SVN is a game-changer



How We Did It: Automation & Testing



[[Home](#) | [Support](#) | [Sales](#) | [Solvers](#) | [Documentation](#) | [Model Library](#) | [Search](#) | [Con](#)]

Latest GAMS System Builds and Test Results

Tuesday 09Nov10 15:

[[Latest Builds](#) | [Alpha Builds](#) | [Beta Builds](#) | [Nightly Builds](#) | [System Codes](#) | [History](#)]

[Cor](#)

nightly α	System	Libraries	Build	Rev	Status and Time (UTC)	Initial Tests	Full Tests
Monday	lnx	Download	23.7.0	21065	Test done 09Nov2010 06:52:24	712 runs 0 failures (q=0,s=0)	Report 8902 runs 2 failures (q=1,s=1)
Monday	lx3	Download	23.7.0	21065	Test done 09Nov2010 09:11:56	732 runs 0 failures (q=0,s=0)	Report 9385 runs 2 failures (q=1,s=1)
Tuesday	vs8	Download	23.7.0	21075	Test done 09Nov2010 14:32:06	734 runs 0 failures (q=0,s=0)	Report 9397 runs 2 failures (q=1,s=1)
Monday	wei	Download	23.7.0	21072	Test done 09Nov2010 07:05:24	682 runs 1 failures (q=1,s=0)	Report 8364 runs 5 failures (q=2,s=3)
nightly β	System	Libraries	Build	Rev	Status and Time (UTC)	Initial Tests	Full Tests
Monday	lnx	Download	23.6.0	21070	Test done 09Nov2010 07:22:45	712 runs 0 failures (q=0,s=0)	Report 8901 runs 2 failures (q=1,s=1)



How We Did It: Automation & Testing

- SVN, other tools automate builds on all platforms
- Extensive, automated tests
 - Test library (501 models), other libraries (hundreds)
 - Runs over all solvers, some NLP/MIP combinations
 - Recent beta: 17 test machines, each ~ 3K – 10K
 - Collecting, archiving, sharing of test results
- PAVER used to compare to previous versions
 - Helps find outliers (bugs), problem cases
 - <http://www.gamsworld.org/performance/paver/>

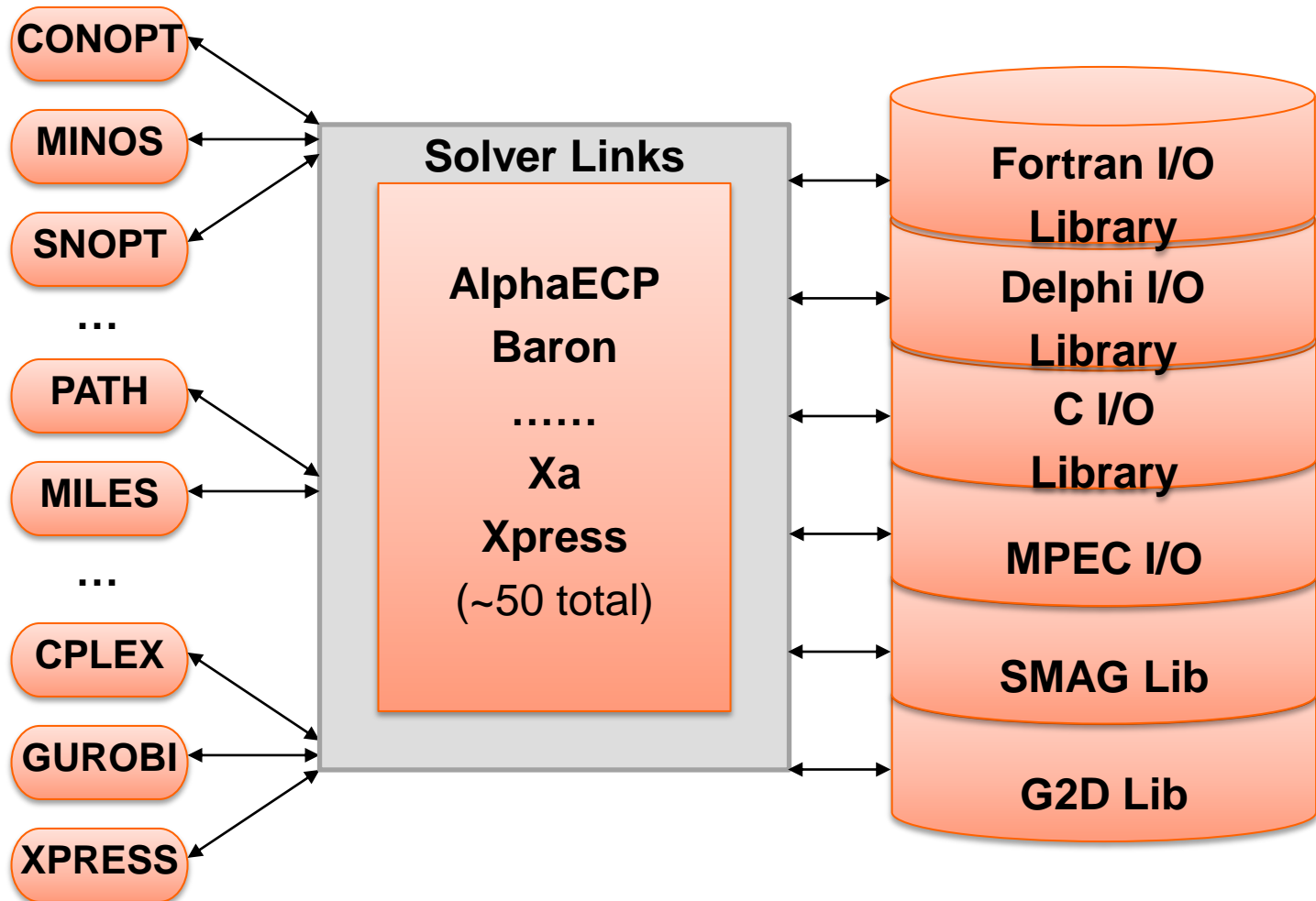


When Will We Be Finished?

- GAMS 23.6.2 (current distribution)
 - Couenne, IPOPT
 - Gurobi
 - OSI-based links to CPLEX, GUROBI, GLPK, MOSEK, XPRESS
 - All previous Fortran links (e.g. CONOPT, MINOS, SNOPT)
 - All links using 2nd-order info (e.g. KNITRO, PATH, MOSEK)
 - Misc others
- GAMS 23.7 – possibly out in April 2011
 - All MCP solvers
 - Meta-solvers like BENCH, Examiner ??

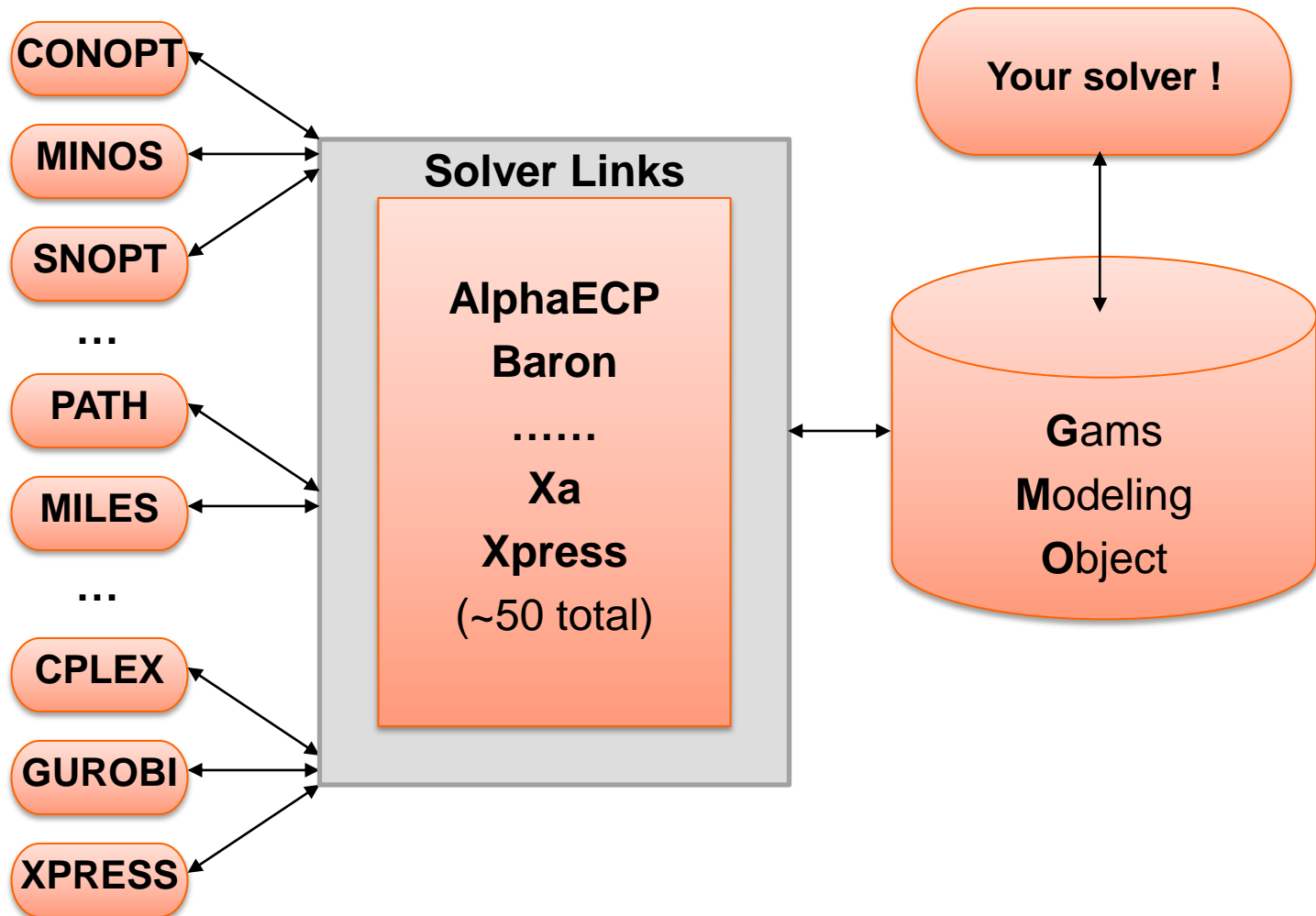


Summary





Summary





Summary

- GMO is part of GAMS distribution
- GMO is used by the majority of GAMS Solver Links - eventually by all
- GMO eases maintenance and makes development process more flexible, more agile
- GMO opens up new possibilities for moving GAMS forward
- GMO interfaces are not yet public but alpha version can be made available on request



Contacting GAMS

Europe

GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany

Phone: +49 221 949 9170

Fax: +49 221 949 9171

<http://www.gams.de>

info@gams.de

USA

GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007
USA

Phone: +1 202 342 0180

Fax: +1 202 342 0181

<http://www.gams.com>

sales@gams.com

support@gams.com