Design Principles that Make the Difference

Franz Nelissen: FNelissen@gams.com
Company Background

Roots: World Bank, 1976

GAMS Development Corporation (Washington)

Tool Provider: General Algebraic Modeling System

Went commercial in 1987

GAMS Software GmbH (Cologne, Braunschweig) 1996
Agenda

Algebraic Modeling Languages – A Success Story

GAMS – Highlights and Design Principles

Model Deployment
The Vision

RESULT:
- Limited drain of resources
- Same representation of models for humans and machines
- Model representation is also model documentation
Algebraic Modeling Languages (AML)

1. High-level computer programming languages
   - Formulation of mathematical optimization problems
   - Notation similar to algebraic notation

2. Do not solve problems directly, but offer links to state-of-the-art algorithms ("solver-links")

Source: http://en.wikipedia.org/wiki/Algebraic_modeling_language
Impact of Algebraic Modeling Languages

1. Simplified model development, changes, and transfer
2. Increased productivity, quality, reliability and maintainability
3. Made a scarce resource (good modelers) more productive

Important vehicle to make mathematical optimization available to a broader audience
2012 INFORMS Impact Prize

Originators of Algebraic Modeling Languages

36 Years later
Agenda

Algebraic Modeling Languages – A Success Story

GAMS – Highlights and Design Principles

Model Deployment
What does he **have to think about?**

1. Problem
2. Mathematics
3. Programming
4. Performance
5. Scalability
6. Connectivity
7. Deployment
8. Maintenance (Life Cycle)
9. ...

**Why use an AML like GAMS?**
General Algebraic Modeling System

Broad User Community and Network

GAMS used in more than 120 countries

25+ Years GAMS Development
General Algebraic Modeling System

Broad User Community and Network

More than 10,000 licenses

50% academic users, 50% commercial users

6,000+ monthly downloads of the free system

25+ Years
GAMS Development
**Broad Range of Application Areas**

<table>
<thead>
<tr>
<th>Agricultural Economics</th>
<th>Applied General Equilibrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical Engineering</td>
<td>Economic Development</td>
</tr>
<tr>
<td>Econometrics</td>
<td>Energy</td>
</tr>
<tr>
<td>Environmental Economics</td>
<td>Engineering</td>
</tr>
<tr>
<td>Finance</td>
<td>Forestry</td>
</tr>
<tr>
<td>International Trade</td>
<td>Logistics</td>
</tr>
<tr>
<td>Macro Economics</td>
<td>Military</td>
</tr>
<tr>
<td>Management Science/OR</td>
<td>Mathematics</td>
</tr>
<tr>
<td>Micro Economics</td>
<td>Physics</td>
</tr>
</tbody>
</table>

**25+ Years**

GAMS Development
Strong Development Environment

GAMS IDE

• Project management
• Editor / Syntax coloring / Spell checks
• Listing file / Tree view / Syntax-error navigation
• Model Debugging / Profiling
• Solver selection / Option selection
• Data viewer
  • Export
  • Charting
• GAMS Process Control
• Model Libraries - 1250 Models included

Everything for rapid model development
Design Principles

1. Simple modeling language with a balanced mix of declarative and procedural elements

2. Open architecture and interfaces to other systems

3. Independent layers

Model
- Platform
- Solver
- Data
- Interface
Simple Declarative Language

1. Language similar to mathematical notation

2. Few basic language elements: sets, parameters, variables, equations, models → Easy to learn

3. Lot’s of code optimization under the hood

Model

Platform  Solver  Data  Interface
Example

Objective

Observe supply limit at plant $i$:
\[ \sum_j c_{i,j} \times x_{i,j} \leq a_i \quad \forall i \]

Satisfy demand at market $j$:
\[ \sum_i x_{i,j} \geq b_j \quad \forall j \]
\[ x_{i,j} \geq 0 \quad \forall i,j \]

Model is executable description of the problem
Mix of Declarative and **Procedural** Elements

**Procedural elements like loops, for, if, macros and functions**

- Allow to build complex problem algorithms within GAMS
- Interaction with other systems:
  - Job control
  - Data exchange

**Combine models inside the language**
Independence of Model and Operating System

Platforms supported by GAMS:

- Models can be moved between platforms with ease!
Independence of Model and Solver

One environment for a wide range of model types and solvers

- All major commercial LP/MIP solver
- Open Source Solver (COIN)
- Also solver for NLP, MINLP, global, and stochastic optimization

Switching between solvers with one line of code!
Independence of **Model and Data**

- **Declarative Modeling**: \( x(j), j \in \{1, \ldots\} \)

- **ASCII**: Initial model development

- **GDX**: Binary Data layer ("contract") between GAMS and applications
  - Platform independent
  - Direct GDX interfaces and general API
Independence of **Model and User Interface**

1. **Open architecture and interfaces to other systems**
   - No preference for a particular user interface

2. **Application Programming Interfaces**
   - *Low Level*
   - *Object Oriented*: .Net, Java, Python, ...

3. **Smart Links to popular environments**
   - Excel, MATLAB, R, ...
Agenda

Algebraic Modeling Languages – A Success Story

GAMS – Highlights and Design Principles

Model Deployment
Is Optimization special?

Observation:
Optimization models
- are expensive to develop
- may have long a lifespan

Modeling Systems & Applications have to be adjusted
- New computer paradigms
- New solver technology and solution methods
- New graphical user interfaces and deployment environments
Change in Focus: Past

**Computation**
Users: Left out

**Model**
Users: Involved

**Application**
Users: Not aware of model
Change in Focus: Now

Computation Users: → Left out

Model Users: → Involved

Application Users: → Not aware of model
Change in Focus: **Now / Future**

**Computation**
Users: ➔ **Left out**

**Model**
Users: ➔ **Involved**

**Application**
Users: ➔ **Not aware of model**
Change in Focus: Modeler...

- Small Community: 2010 ~ 64,000 OR Analytic Professionals in the US
Change in Focus: **Application Developer**

- Software Architecture, Object Oriented Design
- Components, Encapsulation, Classes, Data Access Layer, ...
- Agile Programming, Mesh, ...

- Rapidly changing IT environments
Example – All in One – Top Down

- Add “AML” to existing analytical software system
- “large” user base, e.g. MATLAB, or SAS
Example – All in One – Bottom Up

- Integrate GUI-builder into AML
- “small” user base, e.g. AIMMS (Pro) or FICO Xpress-Insight
Example – Composite Application

- “Construction Kit” with different connected elements
- Use (open source) existing framework to build applications, e.g. IBM ODME

AML

Database

Analytical Software

Application Framework
e.g. Eclipse (IBM), NetBeans (Oracle), .NET (Microsoft)
Summary

AML – A Success Story

Design Principles

- Simple, but powerful language
- Open interfaces
- Different layers

Model Deployment

- Is optimization special?
- Provide cutting edge technology
- Don’t lock developers and users into a certain environment