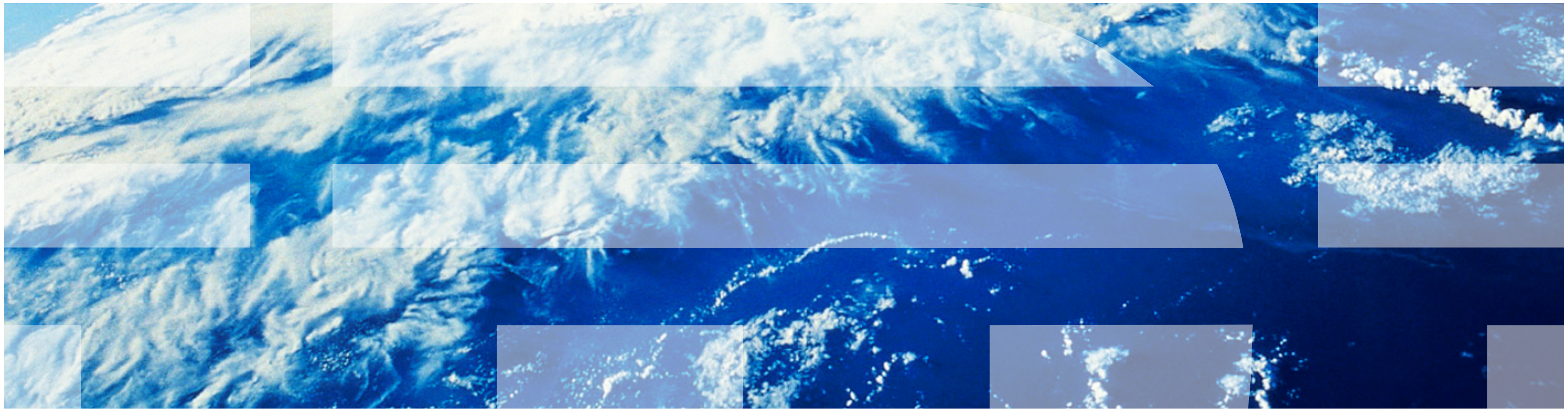
# Optimization : from mathematical tools to real applications

## Illustration on IBM ILOG Optimization technologies and solutions

Philippe Laborie (laborie@fr.ibm.com)



December 12, 2013

## Optimization : from mathematical tools to real applications

- IBM ILOG Optimization technologies and solutions
- From mathematical tools to real applications

## IBM ILOG Optimization technologies and solutions

- Optimization
  - Let:

| | |
|---|---|
| • `X` | A vector of decision variables |
| • `f(X)` | A function |
| • `C(X)` | Some constraints limiting the possible combinations of values for decision variables |

  - Optimization problem:

```
minimize f(X)
subject to C(X)
```

## IBM ILOG Optimization technologies and solutions

- Optimization: **Linear Programming**
  - Let:

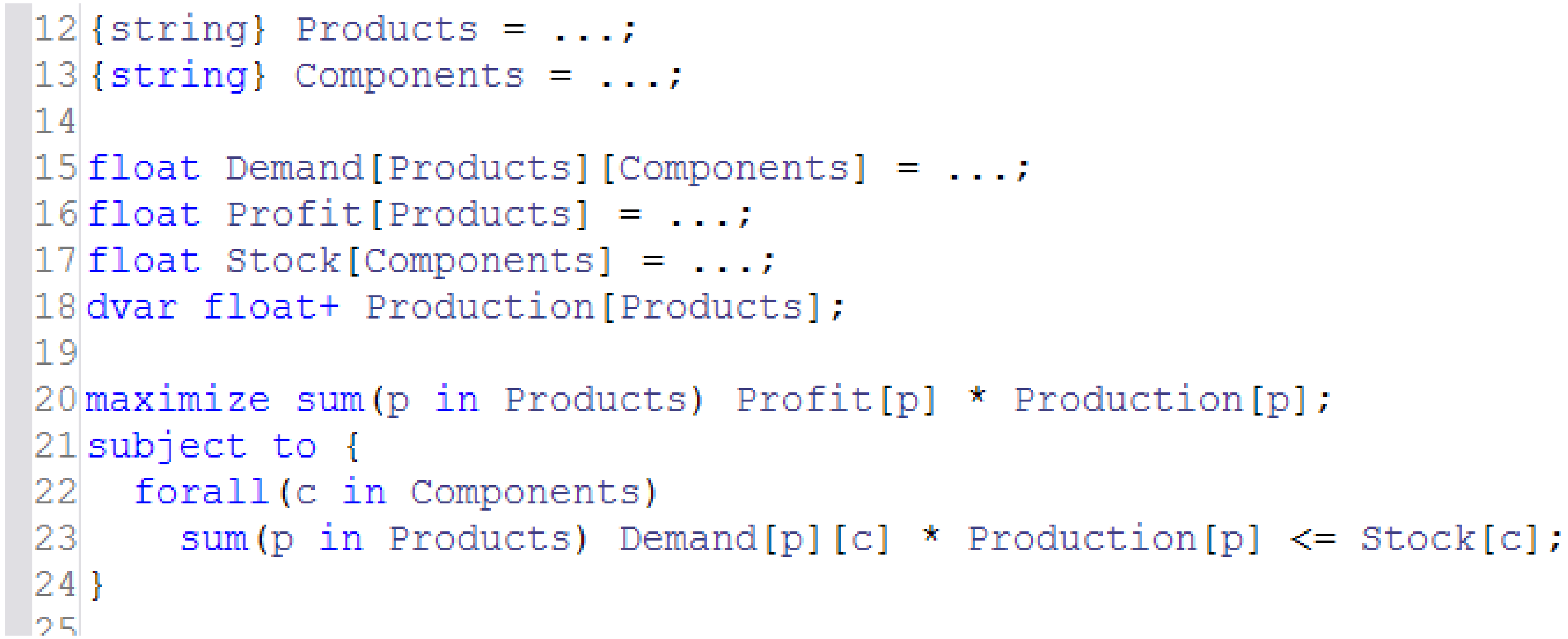
| | |
|---|---|
| **`X`** | A vector of **numerical** decision variables |
| **`f(X)`** | A **linear** function |
| **`C(X)`** | Some **linear** constraints limiting the possible combinations of values for decision variables |

  - Optimization problem:

```
minimize f(X)
subject to C(X)
```

## IBM ILOG Optimization technologies and solutions

- Optimization: **Linear Programming**



```
{string} Products = ...;
{string} Components = ...;

float Demand[Products][Components] = ...;
float Profit[Products] = ...;
float Stock[Components] = ...;
dvar float+ Production[Products];

maximize sum(p in Products) Profit[p] * Production[p];
subject to {
  forall(c in Components)
    sum(p in Products) Demand[p][c] * Production[p] <= Stock[c];
}
```

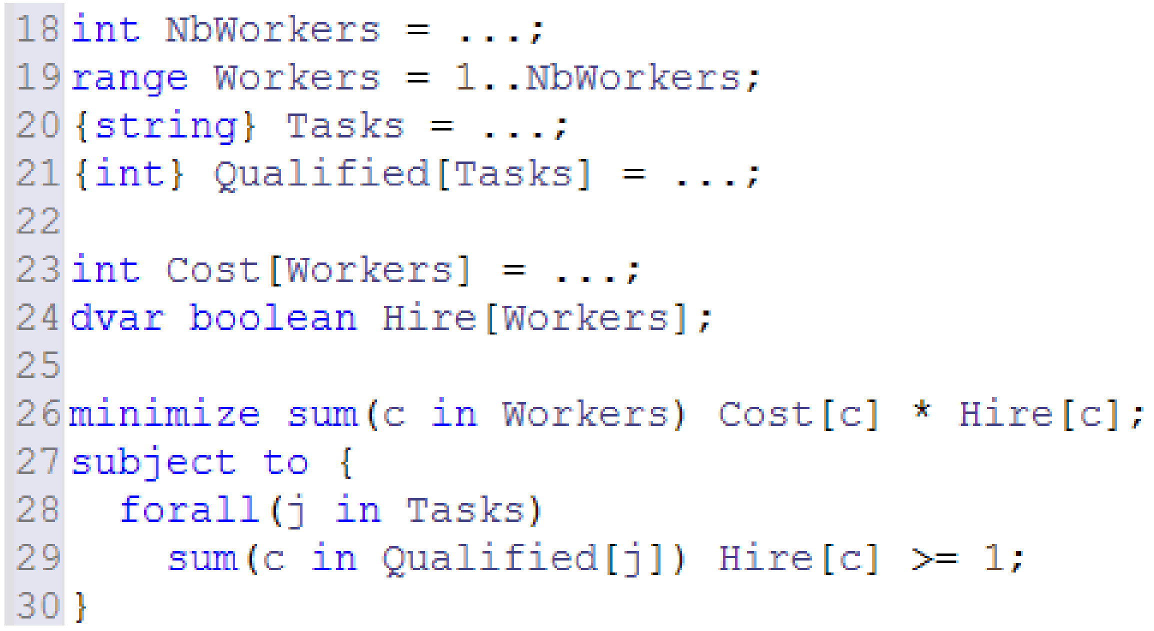
## IBM ILOG Optimization technologies and solutions

- Optimization: **Mixed Integer Linear Programming**
  - Let:
    - **`X`** A vector of **numerical or integer** decision variables
    - **`f(X)`** A **linear** function
    - **`C(X)`** Some **linear** constraints limiting the possible combinations of values for decision variables
  - Optimization problem:

```
minimize f(X)
subject to C(X)
```

## IBM ILOG Optimization technologies and solutions

- Optimization: **Mixed Integer Linear Programming**



```
int NbWorkers = ...;
range Workers = 1..NbWorkers;
{string} Tasks = ...;
{int} Qualified[Tasks] = ...;

int Cost[Workers] = ...;
dvar boolean Hire[Workers];

minimize sum(c in Workers) Cost[c] * Hire[c];
subject to {
  forall(j in Tasks)
    sum(c in Qualified[j]) Hire[c] >= 1;
}
```

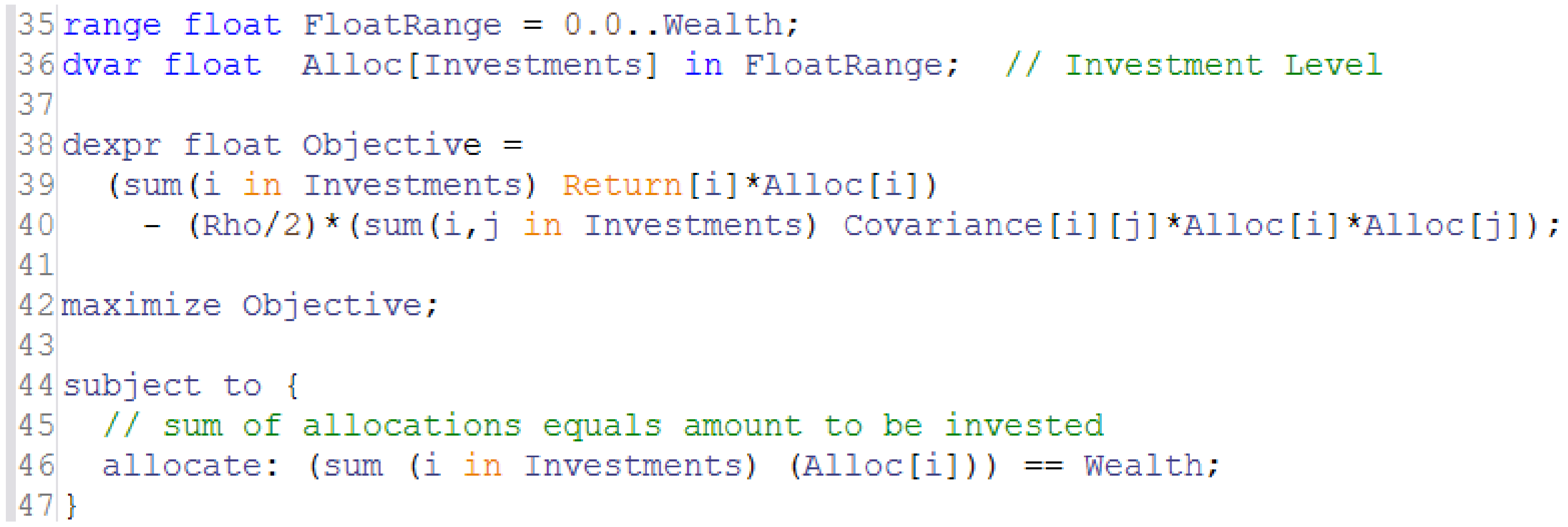
# IBM ILOG Optimization technologies and solutions

- Optimization: **Quadratic Programming**
  - Let:
    - **`X`** A vector of **numerical or integer** decision variables
    - **`f(X)`** A **quadratic** or **linear** function
    - **`C(X)`** Some **quadratic** or **linear** constraints limiting the possible combinations of values for decision variables
  - Optimization problem:

```
minimize f(X)
subject to C(X)
```

# IBM ILOG Optimization technologies and solutions

- Optimization: **Quadratic Programming**



```
range float FloatRange = 0.0..Wealth;
dvar float  Alloc[Investments] in FloatRange;  // Investment Level

dexpr float Objective =
  (sum(i in Investments) Return[i]*Alloc[i])
    - (Rho/2)*(sum(i,j in Investments) Covariance[i][j]*Alloc[i]*Alloc[j]);

maximize Objective;

subject to {
  // sum of allocations equals amount to be invested
  allocate: (sum (i in Investments) (Alloc[i])) == Wealth;
}
```

## IBM ILOG Optimization technologies and solutions

- Optimization: **Constraint Programming**

  – Let:

  | | |
  |---|---|
  | • **`X`** | A vector of **integer** decision variables |
  | • **`f(X)`** | A **general expression** |
  | • **`C(X)`** | Some **general** constraints limiting the possible combinations of values for decision variables |

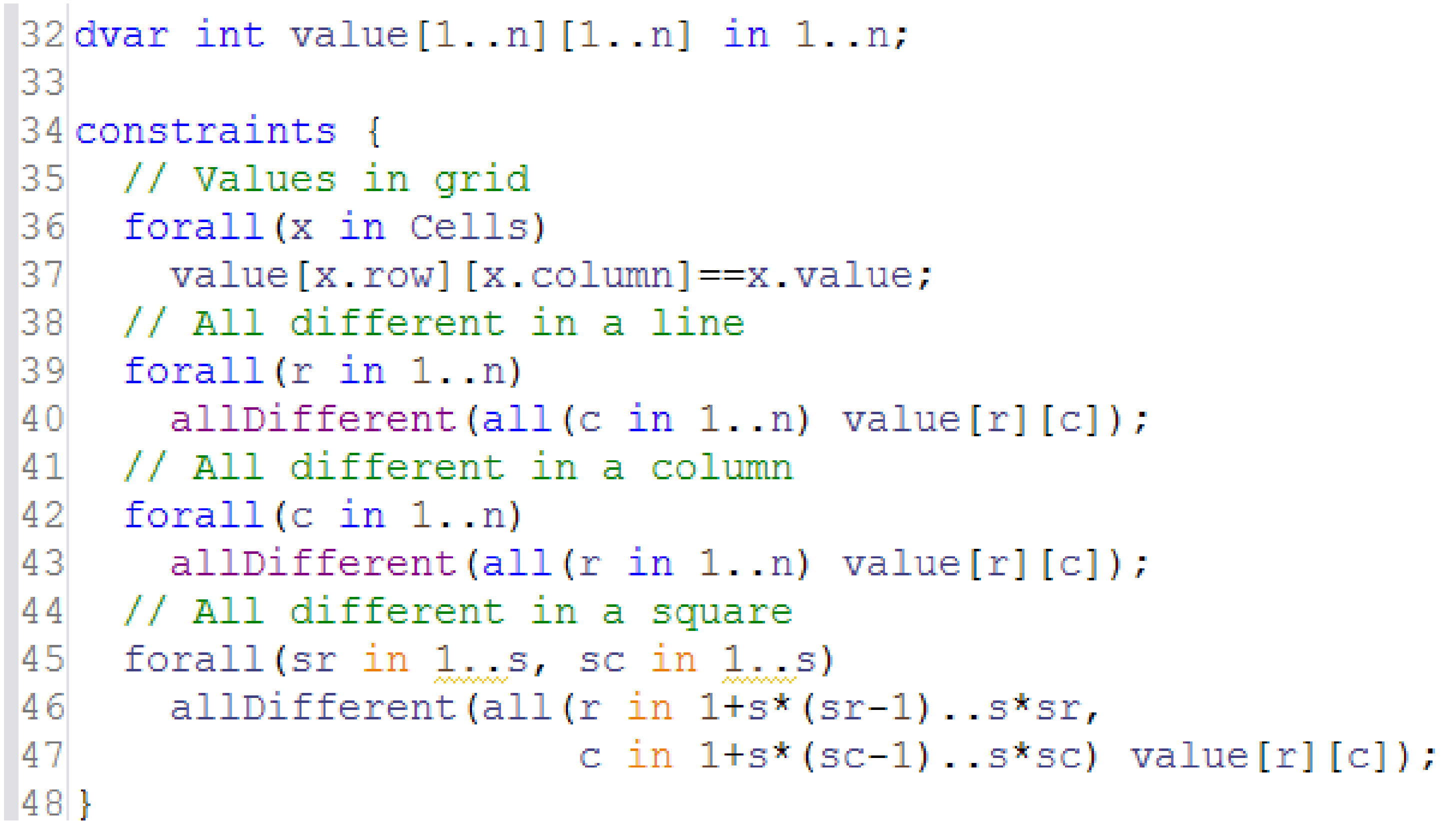
  – Optimization problem:

```
minimize f(X)
subject to C(X)
```

IBM

## IBM ILOG Optimization technologies and solutions

- Optimization: **Constraint Programming**



```
dvar int value[1..n][1..n] in 1..n;

constraints {
  // Values in grid
  forall(x in Cells)
    value[x.row][x.column]==x.value;
  // All different in a line
  forall(r in 1..n)
    allDifferent(all(c in 1..n) value[r][c]);
  // All different in a column
  forall(c in 1..n)
    allDifferent(all(r in 1..n) value[r][c]);
  // All different in a square
  forall(sr in 1..s, sc in 1..s)
    allDifferent(all(r in 1+s*(sr-1)..s*sr,
                     c in 1+s*(sc-1)..s*sc) value[r][c]);
}
```

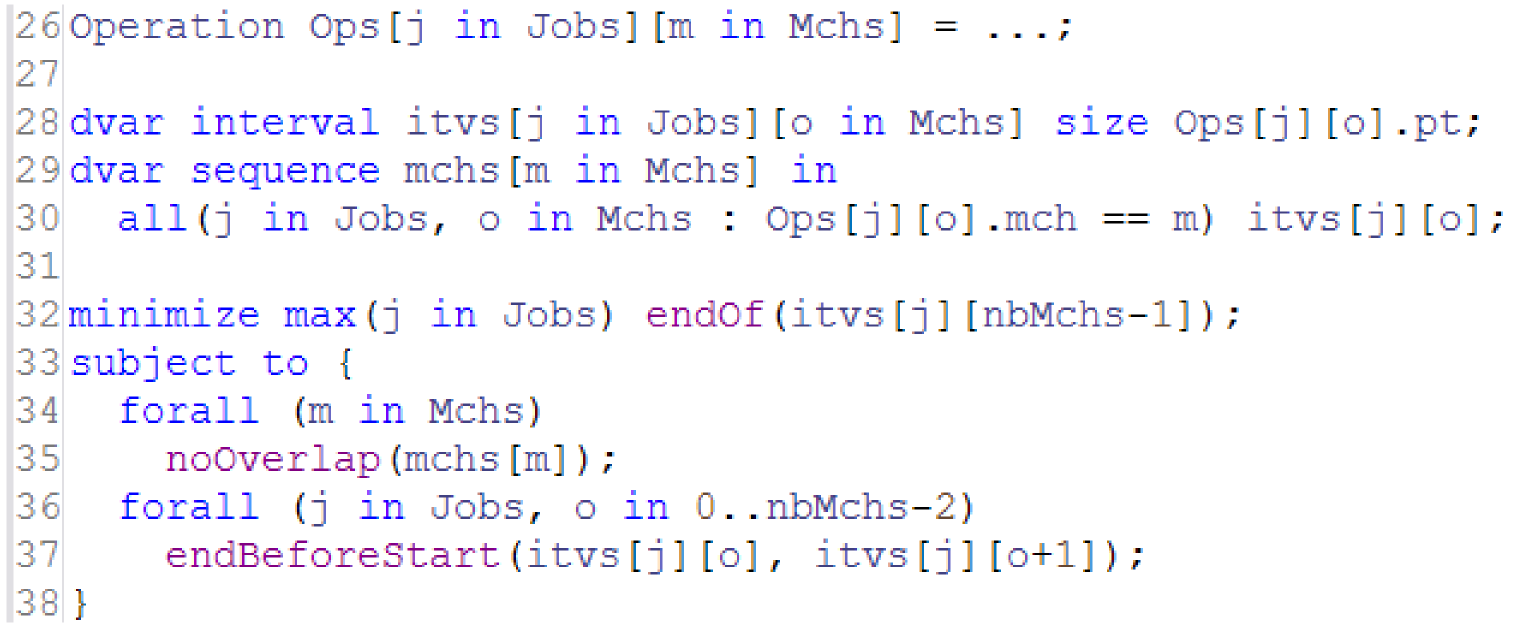
## IBM ILOG Optimization technologies and solutions

- Optimization: **Constraint-Based Scheduling**
  - Let:
    - `X` A vector of **integer** and **interval** decision variables (**activities**)
    - `f(X)` A **general expression**
    - `C(X)` Some **general** constraints limiting the possible combinations of values for decision variables (**temporal constraints, resources**, …)
  - Optimization problem:

```
minimize f(X)
subject to C(X)
```

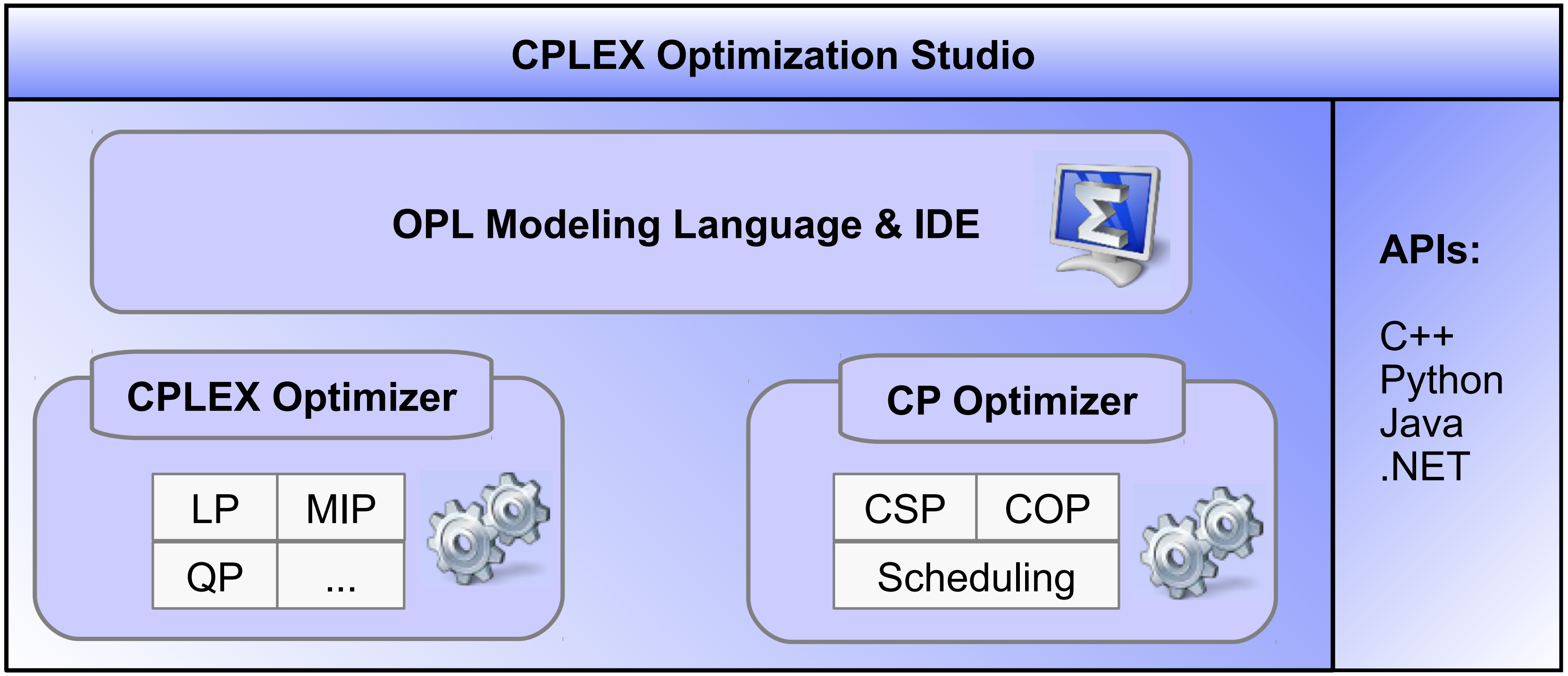
## IBM ILOG Optimization technologies and solutions

- Optimization: **Constraint-Based Scheduling**

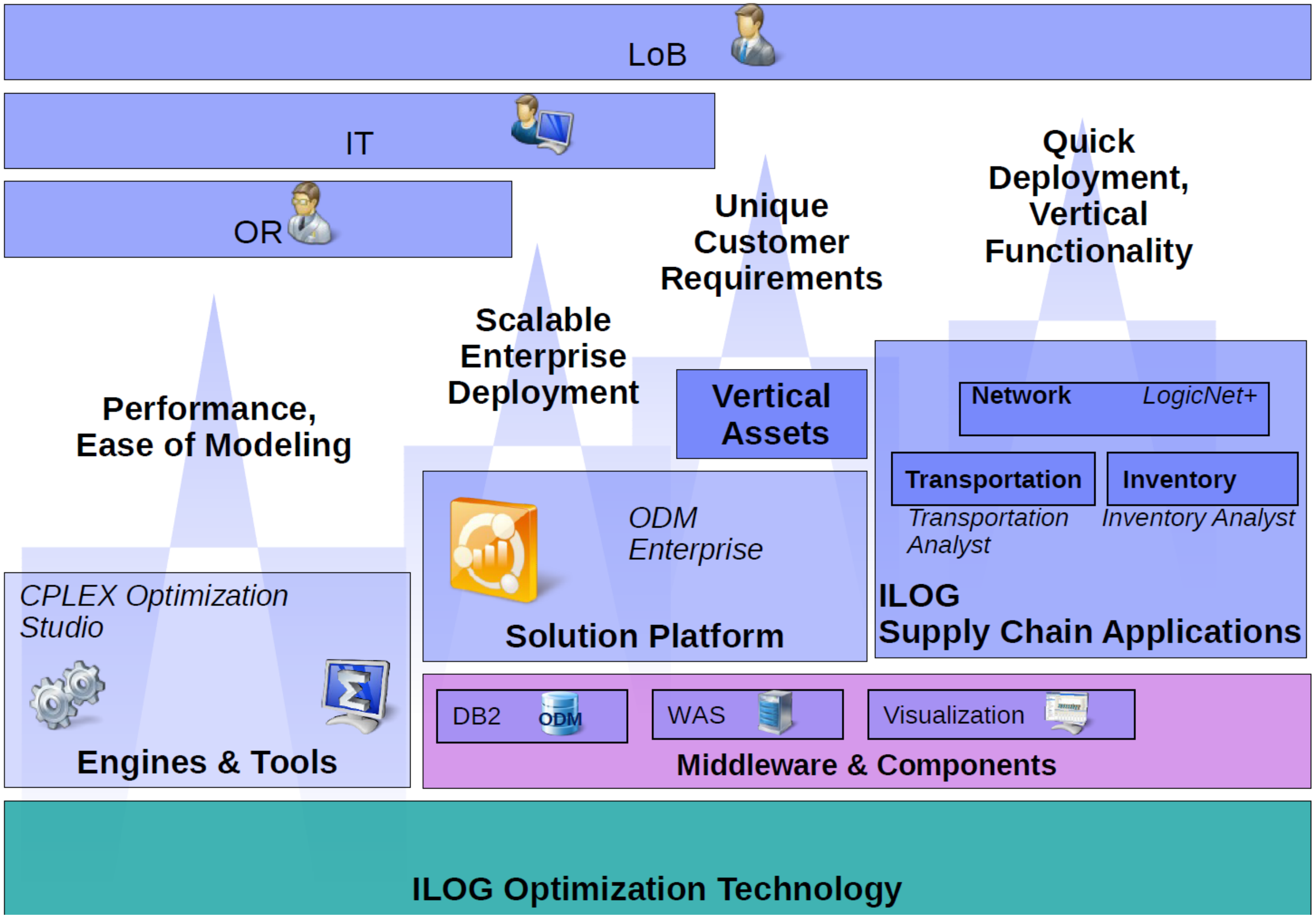


```
Operation Ops[j in Jobs][m in Mchs] = ...;

dvar interval itvs[j in Jobs][o in Mchs] size Ops[j][o].pt;
dvar sequence mchs[m in Mchs] in
  all(j in Jobs, o in Mchs : Ops[j][o].mch == m) itvs[j][o];

minimize max(j in Jobs) endOf(itvs[j][nbMchs-1]);
subject to {
  forall (m in Mchs)
    noOverlap(mchs[m]);
  forall (j in Jobs, o in 0..nbMchs-2)
    endBeforeStart(itvs[j][o], itvs[j][o+1]);
}
```

# IBM ILOG Optimization technologies and solutions



# IBM ILOG Optimization technologies and solutions

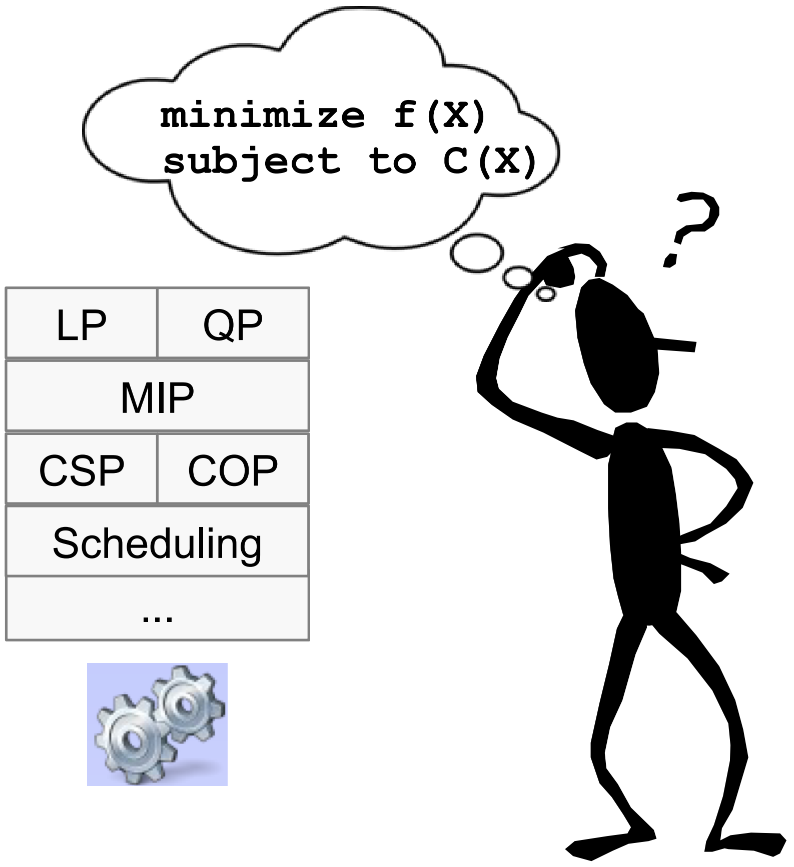

LoB
IT
OR
Performance, Ease of Modeling
Scalable Enterprise Deployment
Unique Customer Requirements
Quick Deployment, Vertical Functionality
Vertical Assets
Network
LogicNet+
Transportation
Inventory
Transportation Analyst
Inventory Analyst
ILOG Supply Chain Applications
ODM Enterprise
Solution Platform
CPLEX Optimization Studio
Engines & Tools
DB2
ODM
WAS
Visualization
Middleware & Components
ILOG Optimization Technology

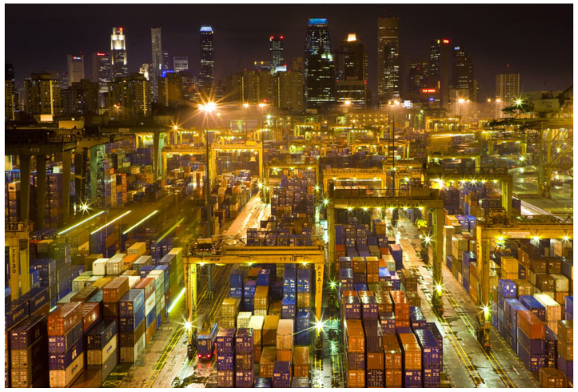
## From mathematical tools to real applications

- Some real applications
  - Aerospace
    - Project scheduling, Aircraft assembly scheduling, Assembly line configuration, …
  - Banking, Finance
    - Portfolio optimization, Trade matching and timing, Cash management, Loan configuration and lending, …
  - Energy & Utilities
    - Network planning, Nuclear plant outage scheduling, Maintenance, Production planning, …
  - Logistics, Supply Chain
    - Routing, Planning and scheduling, Shipment planning and order fulfillment, …
  - Manufacturing
    - Network optimization, Production planning and scheduling, Factory configuration, …
  - Media & Communications
    - Advertizing and program scheduling, event and personnel scheduling, …
  - Retail
    - Inventory optimization, Retail distribution and replenishment planning, Store optimization, …
  - Travel and transportation
    - Crew scheduling, Timetabling, Airport resources allocation, Container management, …

# From mathematical tools to real applications

- There is a gap between mathematical tools and the real applications





## From mathematical tools to real applications

- There is a gap between mathematical tools and the real applications

- This presentation uses material from the excellent blog of JF Puget:

  **IT Best Kept Secret Is Optimization**
  https://www.ibm.com/developerworks/mydeveloperworks/blogs/jfp/

## From mathematical tools to real applications

- There is a gap between mathematical tools and the real applications
  - Different stakeholders / objectives
  - Language, Vocabulary
  - Problem definition: which problem(s) do we want to solve?
  - Data issues
  - Solution definition: what is a solution to the problem(s) ?
  - Solution display & User’s interactions
  - Customer’s expectations
  - Selecting the adequate solving technologies
  - Development tools
  - Communication channels with customer
  - Proof of Concepts (POCs)

# From mathematical tools to real applications
# > Different stakeholders / objectives

- Buyer (management)
  - Objectives: maximize ROI, minimize risk
- Final users
  - Objectives: have the business problem solved, minimize change

➢ Their objective are not necessarily the same, they may even conflict

➢ They do not care about the technology
  - They do not understand it, sometimes they do not trust it!
  - They will typically expect that the resolution time of their problem scales linearly with its size

➢ They only care about **<u>their problem</u>**

## From mathematical tools to real applications
## > Different stakeholders / objectives

- Buyer (management)
  - Objectives: maximize ROI, minimize risk
- Final users
  - Objectives: have the business problem solved, minimize change

➢ They only care about **<u>their problem</u>**

➢ … and the integration of the new solution in their environment
  - Can be **very** hard if it implies process changes
  - Can be tough if it implies to move or fire people
  - Easier when optimization is used to do more :
    - More revenue, better service, new services, etc

## From mathematical tools to real applications
## > Different stakeholders / objectives

- Optimization specialists
- Application developers, IT

➢ They deliver the solution to the business problem

➢ Applications developers and IT specialists generally do not know optimization technologies

➢ This is something we are pushing in IBM: make optimization technologies easier so that they are accessible to IT specialists and developers (Model&Run paradigm, ODME)

## From mathematical tools to real applications
## > Different stakeholders / objectives

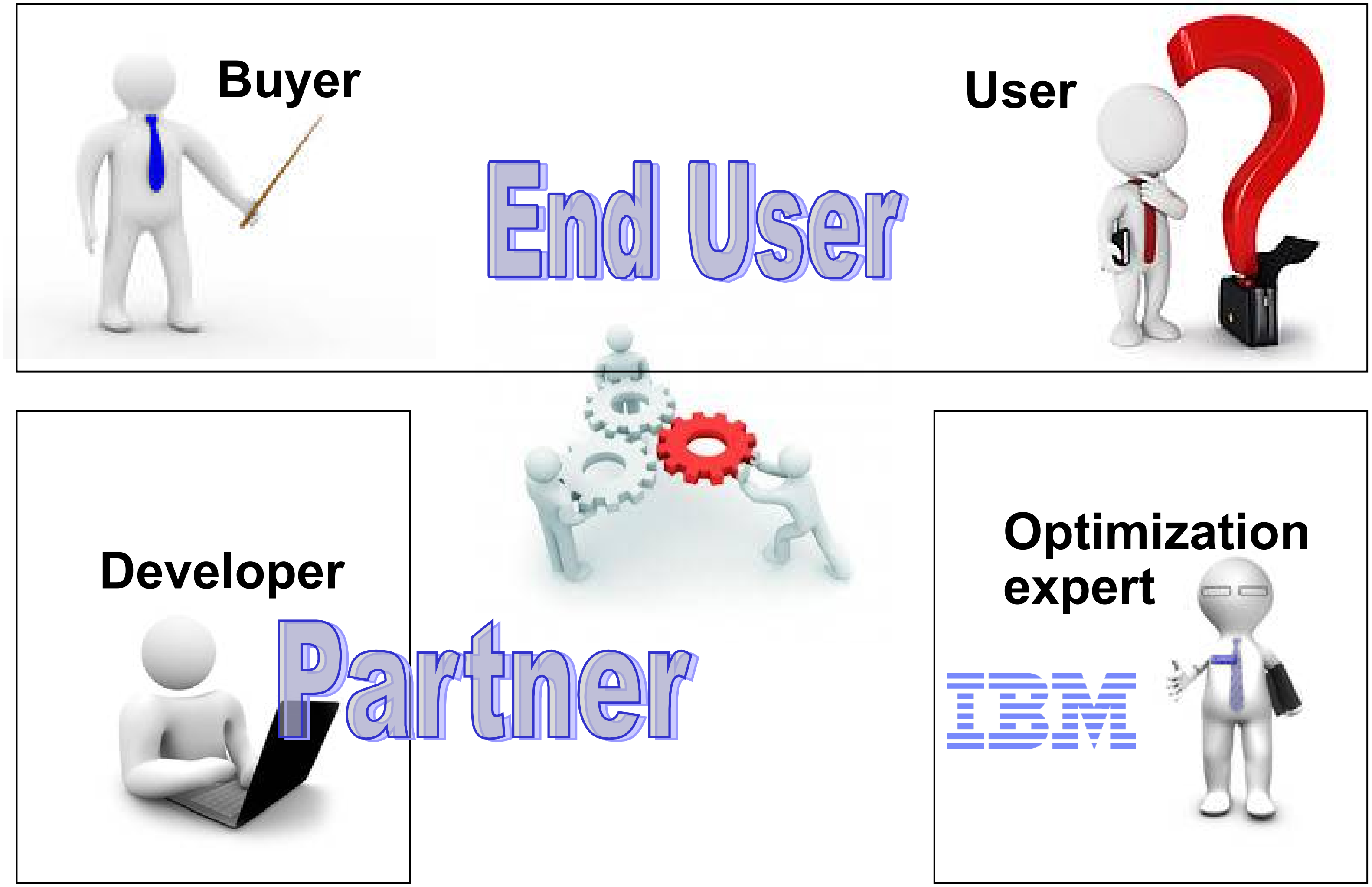
Buyer

User

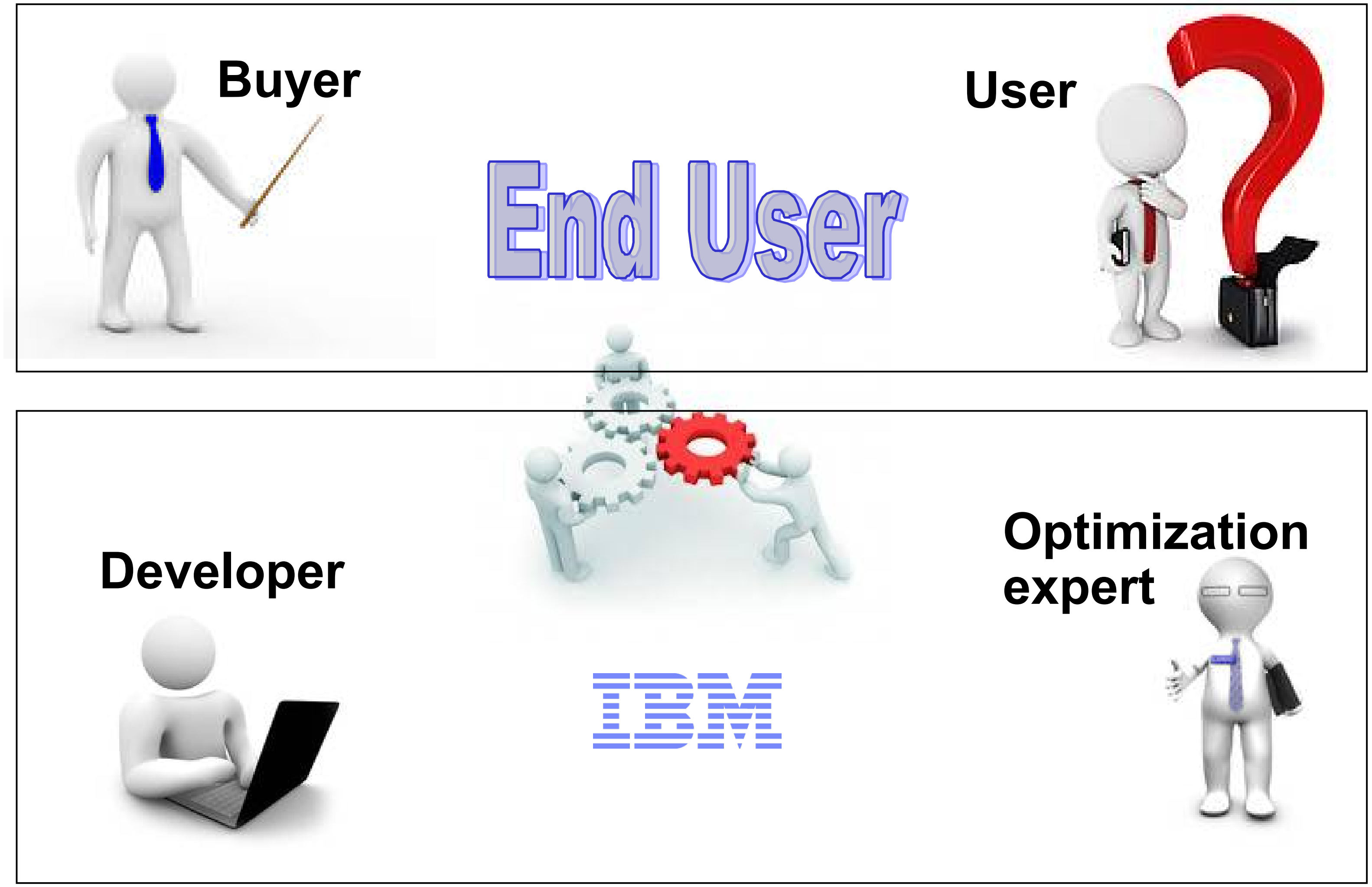
Developer

Optimization expert

## From mathematical tools to real applications
## > Different stakeholders / objectives



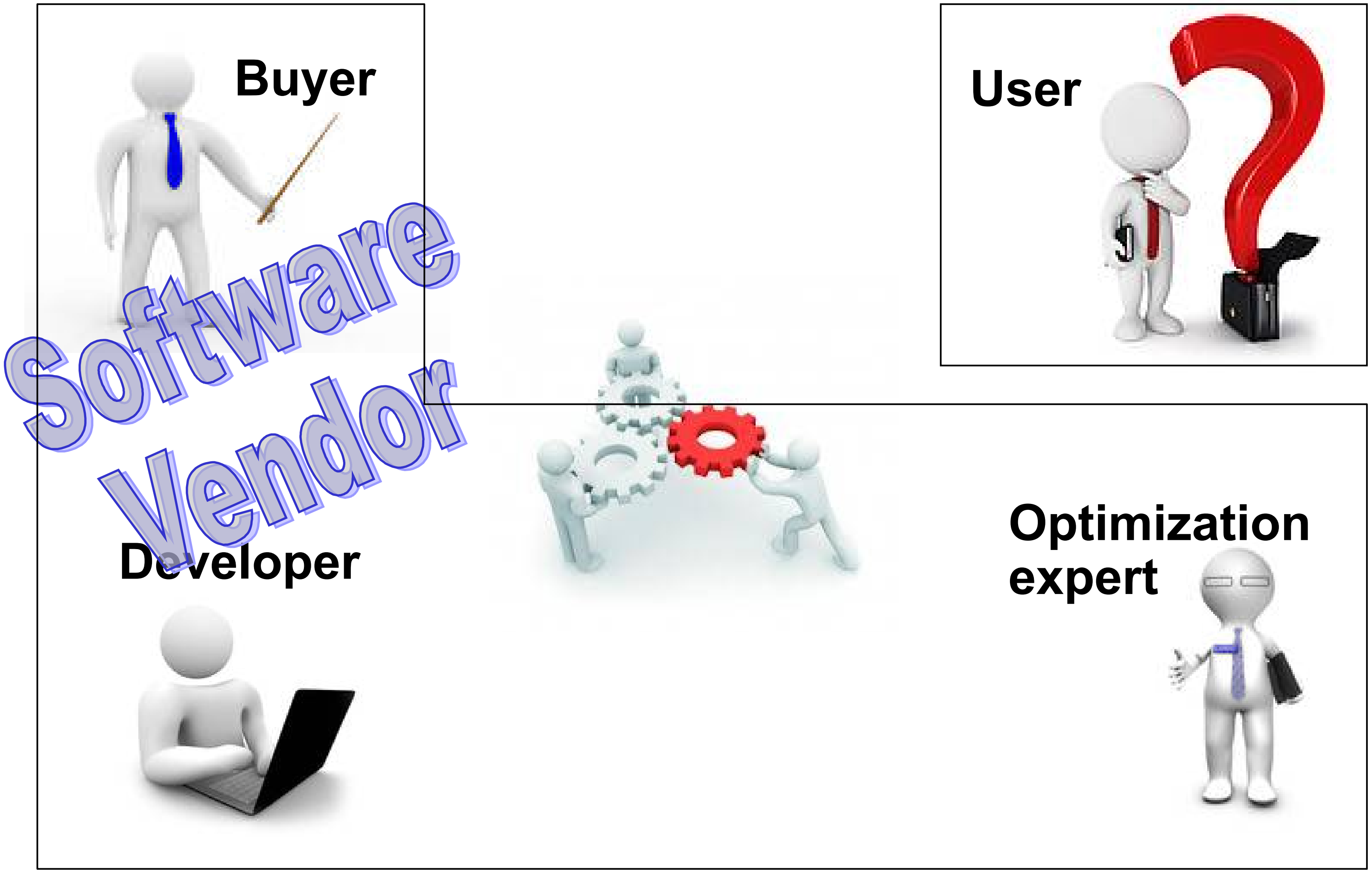
## From mathematical tools to real applications
## > Different stakeholders / objectives



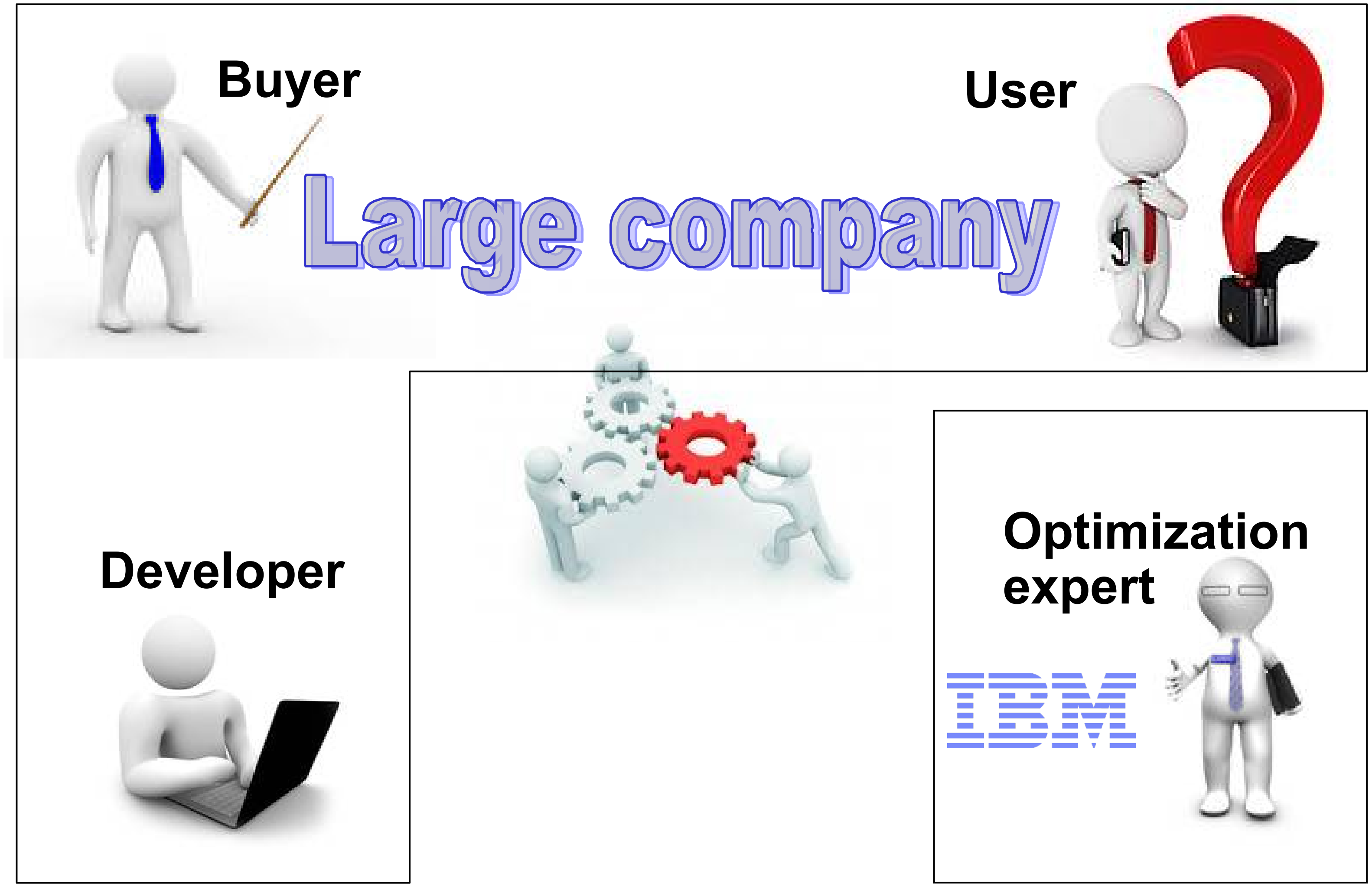
# From mathematical tools to real applications
# > Different stakeholders / objectives

Buyer
Software Vendor
Developer

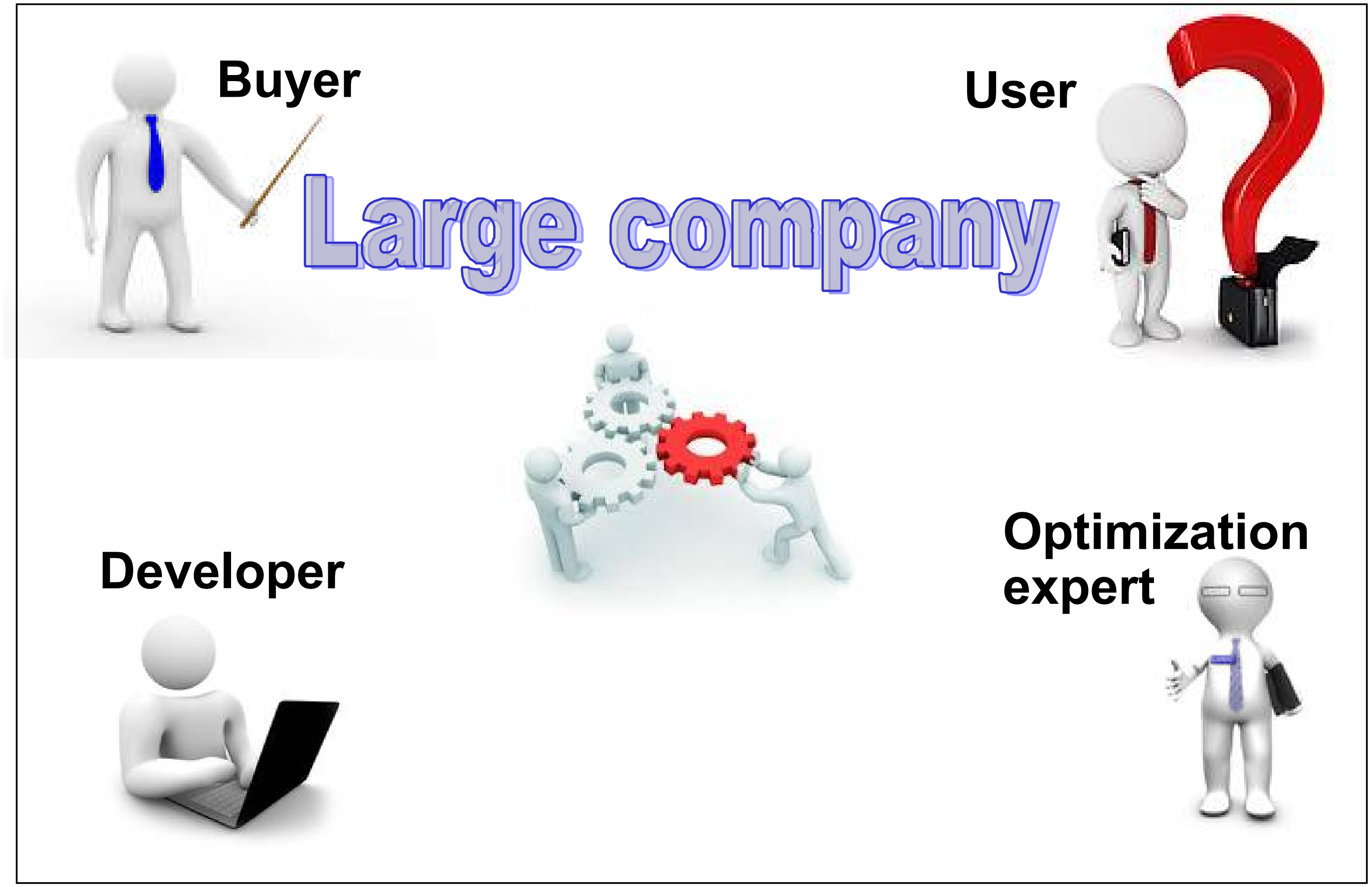
## From mathematical tools to real applications
## > Different stakeholders / objectives



# From mathematical tools to real applications > Different stakeholders / objectives



# From mathematical tools to real applications
# > Different stakeholders / objectives



## From mathematical tools to real applications
## > Different stakeholders / objectives

- You need to have a good understanding of the **ecosystem** of the project

## From mathematical tools to real applications
## > Language, Vocabulary

- Each domain has its own terminology : don't expect a customer to describe his optimization problems in terms of variables, constraints and objective function
- You will hear about: **ingot**, **PSCW**, **APX**, **connection**, **coil**, **berth**, **ratability**, **shelf life**, **crude distillation unit** (CDU), **discount rate**, **reticle**, **WBS**, **palonnier** (fr), **bâti** (fr), …
- You have to communicate with the customer/prospect using his own business language in order to understand his processes

## From mathematical tools to real applications > Problem definition

- What is the problem to be solved?
- Usually, many (most of) constraints and objective functions are **implicit**, they need to be formalized
- A good technique for discovering implicit constraints is to iteratively enrich the model and present some solutions of the current model to the user, he will often easily see:
  - That something is wrong/infeasible in the solution because some **implicit constraint** was not modeled
  - That the solution is not of good quality because some **implicit objective** was not considered
- A good technique for discovering that some constraints have been **overstated** in the model is to match a current solution of the customer against the model and analyze which constraints are violated

## From mathematical tools to real applications
## > Problem definition

- You need to solve the **right problem**
- J.F. Puget: "*It is better to provide an approximate answer to today's problem than to perfectly solve yesterday's problem.*"
- J. Tuckey: "*An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.*"

## From mathematical tools to real applications
## > Problem definition

- **But**: Optimization will need some **model** of the real problem
- Models cannot fully describe the reality, some aspects of the real problem will have to be:
  - Ignored
  - Relaxed (e.g. n almost similar machines => 1 resource of capacity n)
  - Over-constrained (e.g. increase task duration to account for uncertainty)
  - Approximated (e.g. to model an implicit objective function)
- There is a trade-off to be found between:
  - The complexity/precision of the model
  - The capacity of optimization technologies to satisfactorily solve it
  - ➢This is the art of optimization modeling

## From mathematical tools to real applications
## > Problem definition

- Be ready to have **moving targets**
- The user may initially have a limited scope in mind:
  - Focused on a particular sub-problem
  - Think anything large would not be manageable
- … as the project advances, the ambitions may change to widen (or restrict) the scope and/or scale of the model

➢The model should be written in such a way that refactoring is easy (model agility): clean data structures, well-structured code, etc.

➢For instance, even for complex projects, the OPL language makes it natural to separate data manipulation from the heart of the model itself. Furthermore, the heart of the model is in general quite concise.

## From mathematical tools to real applications
## > Data issues

- Data may be stored on different type of source:
  - Data files (txt, csv, excel, …)
  - Data bases
  - Data streams (web, …)
- It can be the result of a computation:
  - Statistical analysis
  - Current state of an execution system
  - …
- Connection to data source should be **easy**

➢IBM Optimization products integrate easy-to-use connectors with data files, data bases, statistic tools like SPSS, Python API …

## From mathematical tools to real applications
## > Data issues

- Data **confidentiality** issues : data may contain strategic information for the user about its processes, its customers, …
- Customer may not agree to share his data as is
- When communicating with the customer, it is useful to have tools / data file formats that encrypt/anonymize the data

➢ CPLEX Optimization Studio 12.6 provides functionalities to save a CP model in an anonymized form (rename variables)

## From mathematical tools to real applications > Data issues

- Having data is not enough, you must have **good quality** data
- Data can be:
  - Uncertain
  - Imprecise
  - Incomplete
  - Incorrect
- Optimization is a GIGO computation (garbage in, garbage out): Wrong data $\Rightarrow$ Useless solutions
- Good practices:
  - Check data quality in the model
  - Visualize solutions to easily see that something is going wrong

## From mathematical tools to real applications
## > Data issues : uncertainties

- When data is uncertain (activity durations, costs, prices, etc.), think of **stochastic optimization** techniques but remember:
- A prerequisite for solving a stochastic optimization problem is that you are able to **solve (efficiently) the deterministic version of the problem**

➢ CPLEX Optimization Studio 12.6 provides a new constraint for modeling stochastic/robust scheduling problems on disjunctive resources

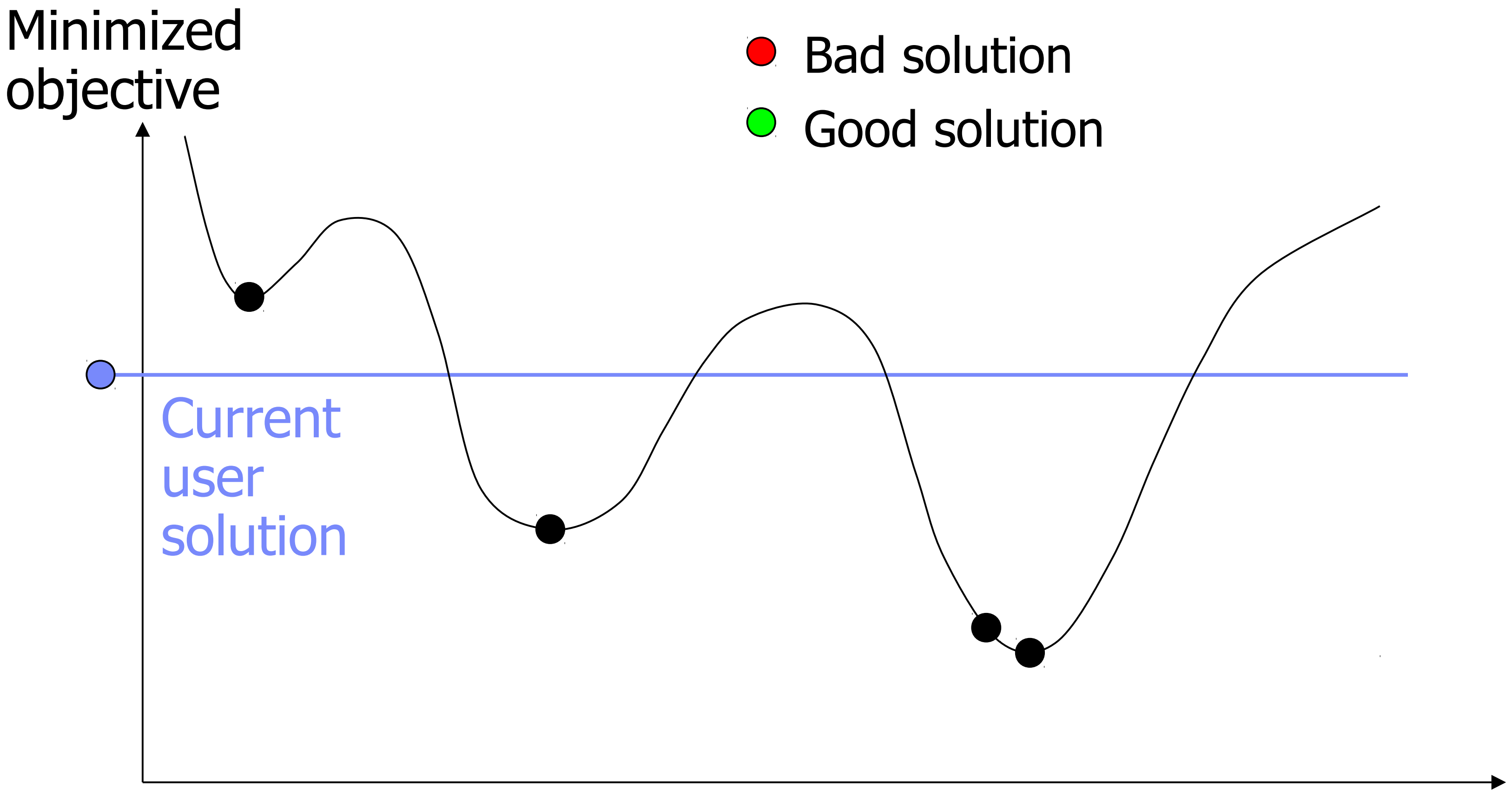
## From mathematical tools to real applications
## > Data issues

- Having good quality data is not enough
- You must get realistic data **early enough** in the project
  - For developing and testing the data structures and the model
  - It helps communication with the customer about his requirements
- You must work with **many instances**, not only one
  - Avoid model over-tuning (see later)

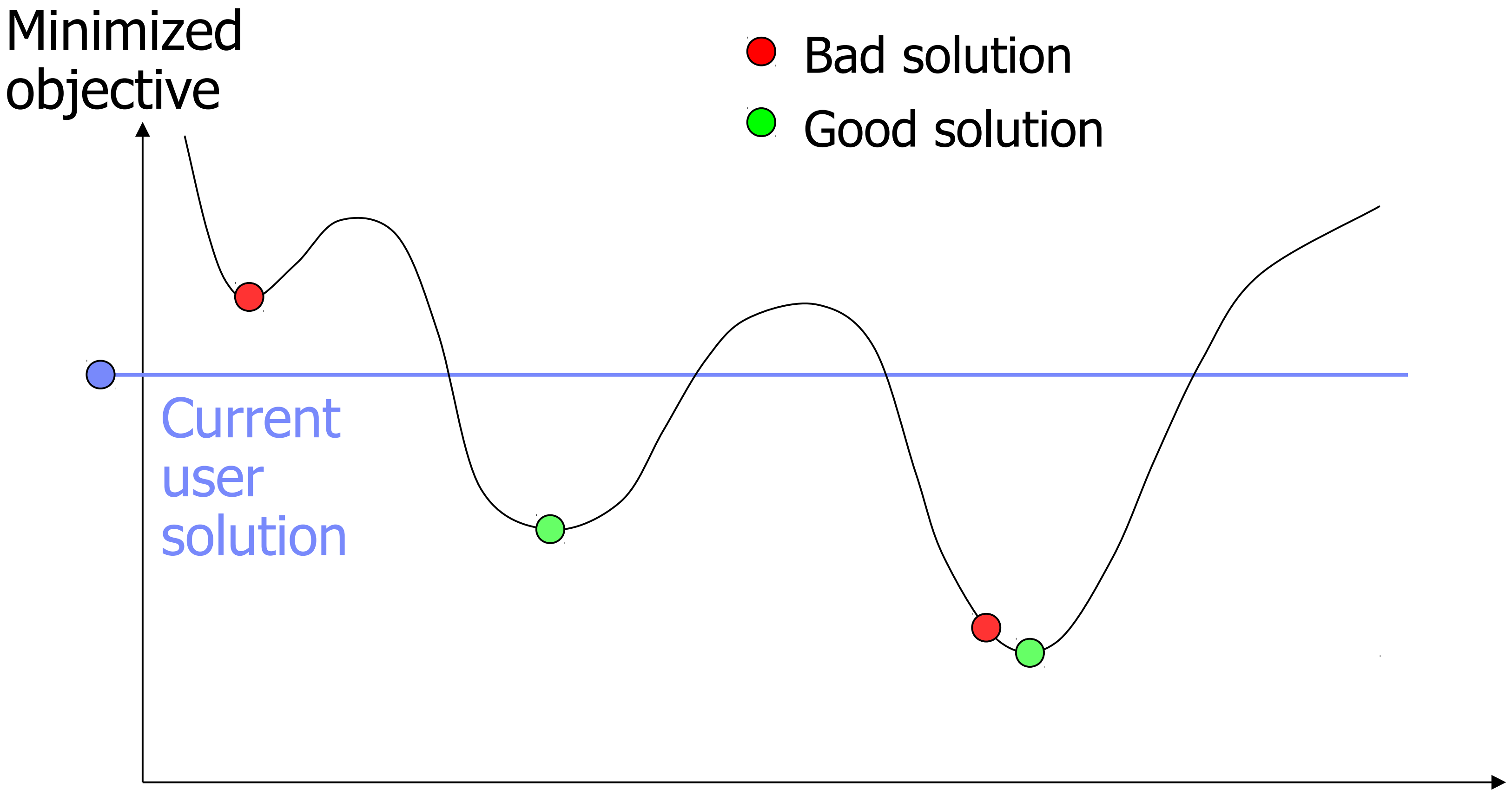
## From mathematical tools to real applications
## > Solution definition

- The notion of **optimal solution** is always relative to a **model** of the real problem not to the real problem itself
- The user only cares about **his real problem**
- If you can convince the customer that the model is close enough to the reality and if you have some guarantees on the solution quality with respect to the model (optimality proof, gap), that can be of course useful **but remember**: the user does not care about the technology and does not understand it
- The user will assess the quality of the solution by:
  - Comparing it to the current solutions he gets
  - Trying to see if the solution you provide cannot be easily improved

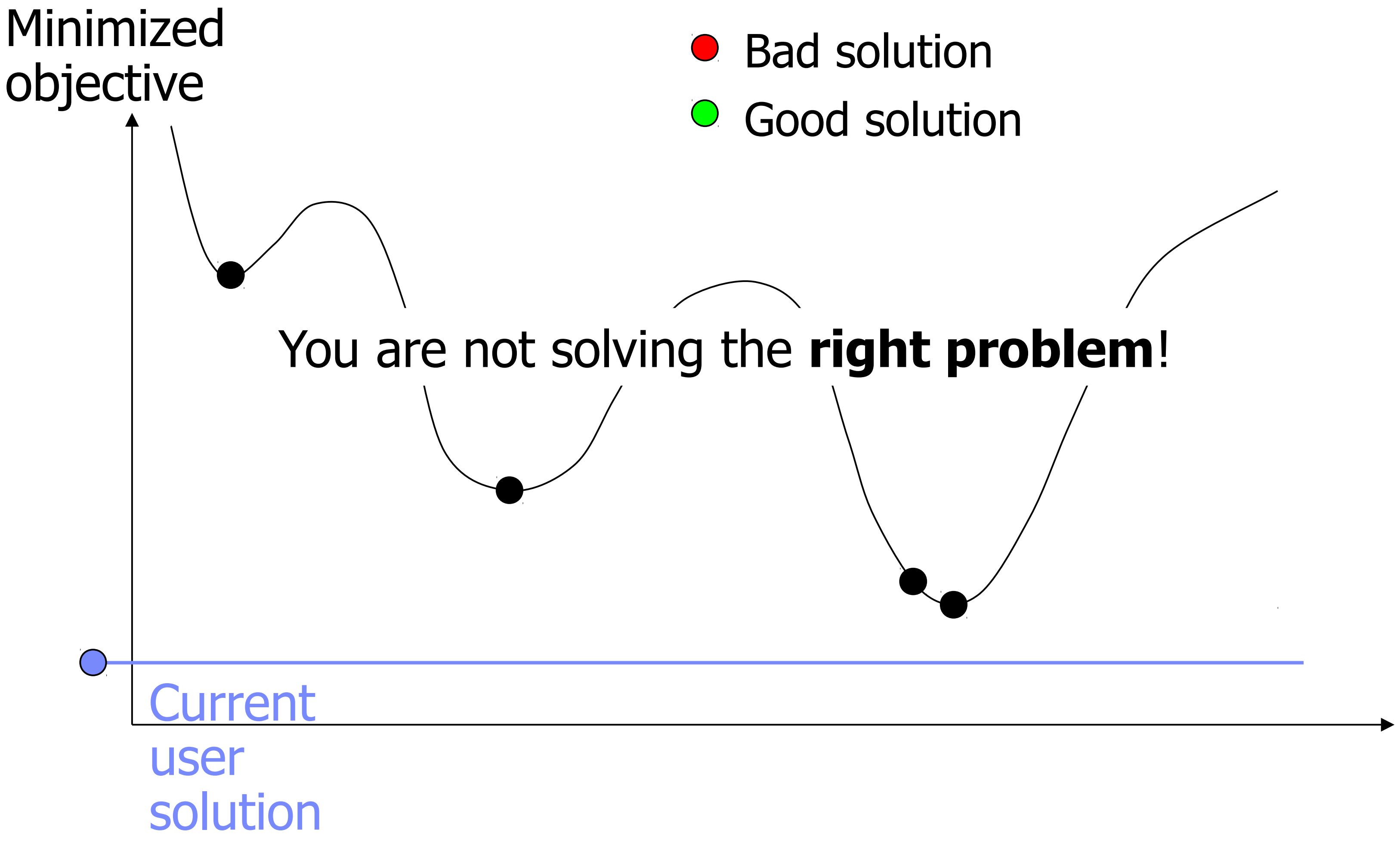
## From mathematical tools to real applications
## > Solution definition



# From mathematical tools to real applications > Solution definition



# From mathematical tools to real applications
> Solution definition



You are not solving the **right problem**!

## From mathematical tools to real applications > Solution definition

- A good solution is a **feasible** and **locally** optimal solution that is **better than the current solution** of the customer

➢ IBM Optimization engines pay special attention to the fact there is no easily improved solution close to the one provided. For instance CPLEX provides the notion of *solution polishing* for that.  CP Optimizer operates on neighborhoods on incumbent solutions.

## From mathematical tools to real applications
## > Solution definition

- **Customers like solutions**
- Instead of one solution, they often prefer to have several ones
  - Because there are several objectives (multi-objective, Pareto optimality)
  - Because he is the “decider”, not the machine

## From mathematical tools to real applications
## > Solution definition

- **Customers like solutions**
- Even when the problem is infeasible, they like to have one solution!
  - Relax constraint in the objective function

➢ CPLEX Optimizer provide the notion of feasopt algorithm to provide a solution that violates the constraints as few as possible

➢ CP Optimizer make it easy to relax constraints as objective terms. Specially for scheduling constraints with the notion of optional interval variable

- … or they would like to understand why the problem is infeasible

➢ CPLEX and CP Optimizers provide a conflict refiner functionality to compute a minimal subset of infeasible constraints

## From mathematical tools to real applications > Solution display / User interactions

- Optimization is not worth it if the results aren't understood by the business users
- The best way to make them understandable is to use **nice graphics**
- It also helps a lot the communication with customer during the development of the model

➢ Solution display and model development should preferably be done in parallel

## From mathematical tools to real applications
## > Solution display / User interactions

- Users often need some “**what if analysis**” and solve different scenarios to analyze the impact of some decisions
  - What happens if I have less trucks available ?
  - What happens if this machine breakdowns ?
  - What happens if the due date of a task is changed ?

➢ This notion of *what if analysis* and scenarios is at the heart of ODME

- **Cooperative** problem solving is also a common paradigm
  - Configuration problems (e.g. car configuration)

## From mathematical tools to real applications > Customer’s expectations

- Requirements on solution architecture & infrastructure (clients, servers, cloud, machines, communication)
- Requirements in terms of response time, scalability
  - Users will in general expect that response time **grows linearly** with problem size
  - … and **decrease linearly** with the number of machine cores
- Requirements in terms of quality
  - **No bugs**. Bugs in the engine may be killers (it's not like for a research work)
  - Solution **determinism**: two resolutions of the same problem in the same conditions should give the same result

## From mathematical tools to real applications
## > Selecting the adequate solving technologies

- LP, MILP, SAT, CP, Scheduling, ... : not always easy to select the adequate technology
- It is a plus if you can easily change from one to the other in case you need to reach a dead-end

➢ IBM Optimization technologies are based on a common modeling layer (called Concert Technology) that makes it easier to change from one optimization technology to another one

## From mathematical tools to real applications
## > Selecting the adequate solving technologies

- Real life problems are often complex:
  - Large size
  - Mix of different source of complexity
- The problem often needs to be split into several optimization models that are solved with different technologies
  - Hierarchical decompositions: planning with MIP, then scheduling with CP, …
  - Temporal decomposition on time-windows
  - Column Generation
  - Logical Benders decompositions
  - …
- Try to understand how the customer is currently solving his problem, it *may* give some directions for decomposing the problem

## From mathematical tools to real applications
## > Selecting the adequate solving technologies

- Model tuning is good
- Model **over-tuning** is **bad** (you will only loose time)
  - Do not fine tune your model until the model is complete
  - Do not fine tune your model on a single problem instance !
  - Do not fine tune your model on a single random seed !

## From mathematical tools to real applications > Development tools

- Which time are you allowed ?
  - Time to **compute** a solution
    - Often time boxed, best solution found in limited time
  - Time to **develop** the software application
    - Boxed too by project funding

- Trade off
  - Fast solver with poor development tool
    - Not much time to tune model/data, poor performance in the end
  - Slow solver with great development tool
    - Lots of time to tune model/data, poor performance in the end
  - Great solver with great development tools
    - Lots of time to tune model/data, great performance in the end

## From mathematical tools to real applications
## > Proof of Concepts (POCs)

- Often, potential customers will require a **POC** to convince them that we are the best for solving their problem

- A POC is the same thing as the final project except that:
  - 1. You have much less time for doing it (a few days/weeks)
  - 2. You are in general not paid for it
  - 3. It will be an essential element for the success of the deal

- The POC is not only about optimization, a good **visualization** and interaction with the user is crucial already at this point

- The POC solves a simpler problem than the actual one. This problem:
  - Must be **relevant** to the customer
  - Must convince the customer (and the model developer) that the approach will scale with problem complexity and with problem size: the POC should work with **real problem size**

## From mathematical tools to real applications > Communication channels with customer

- We need to get feedback on the optimization products and help the costumers
- Different channels for communication with users:
  - Technical sales people
  - Training sessions
  - Consulting (internal, external) $\rightarrow$ Partners
  - Customer support: L1 $\rightarrow$ L2 $\rightarrow$ L3 $\rightarrow$ R&D
  - Discussion forums
  - User's meetings/conferences

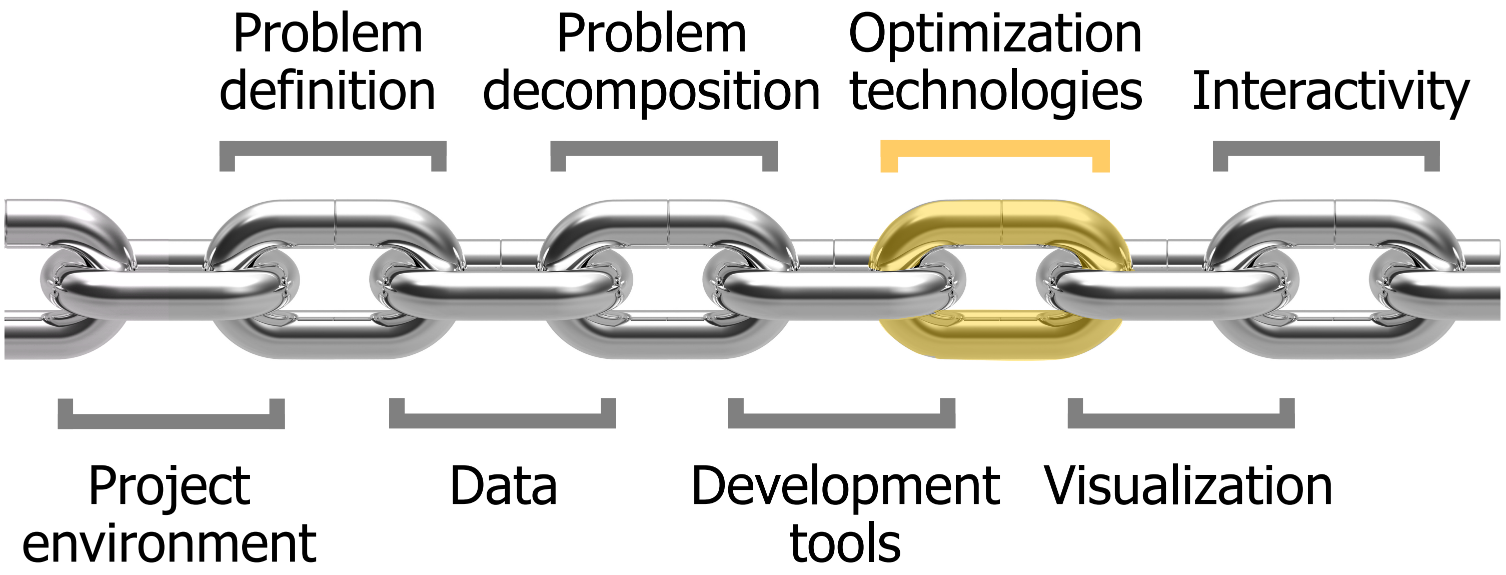
## From mathematical tools to real applications > Take-home messages

- You need to have a good understanding of the **ecosystem** of the project
- You have to understand the **vocabulary** of the customer
- You need to solve the **right problem**
- You need to make sure you access **good quality** data **early** enough
- A good solution is a **feasible** and **locally** optimal solution that is **better than the current solution** of the customer
- Customers **like** solutions (even when the problem is infeasible)
- **Nice graphics** are necessary and should preferably be developed **in parallel** with the model

## From mathematical tools to real applications > Take-home messages

- Optimization is usually not a one-step process, it often requires **interactions** with the user (what if analysis, cooperative problem solving)
- Be ready to **combine several** optimization **models** that are solved with **different technologies**. Be **creative** !
- Model tuning is good, model **over-tuning** is **bad**
- You need **good model development tools**

## From mathematical tools to real applications
## > Take-home messages


Problem definition
Problem decomposition
Optimization technologies
Interactivity
Project environment
Data
Development tools
Visualization