Aol

Microsoft Visual Studio

IBM

irish dev

software SKILLNET
learn, share, grow

dediserve

unum
Better benefits at work.

DNM TECHNOLOGY
POWERING YOUR POTENTIAL

slattery communications

qms
qms-software.com

epicenter

The Irish Software Show 2010

neon sms

Cincom

eset

cloud ICT

Brightwater
recruitment specialists

Cinfinity

Digital Mines
Utility Computing

sagepay

sonru.com

PopCap

WIBU SYSTEMS

clarify
communications consulting ltd

black marble

Java Constraint Programming with JSR-331

Jacob Feldman, PhD
OpenRules Inc., CTO
jacobfeldman@openrules.com
www.openrules.com
www.4c.ucc.ie

epicenter
The Irish Software Show 2010

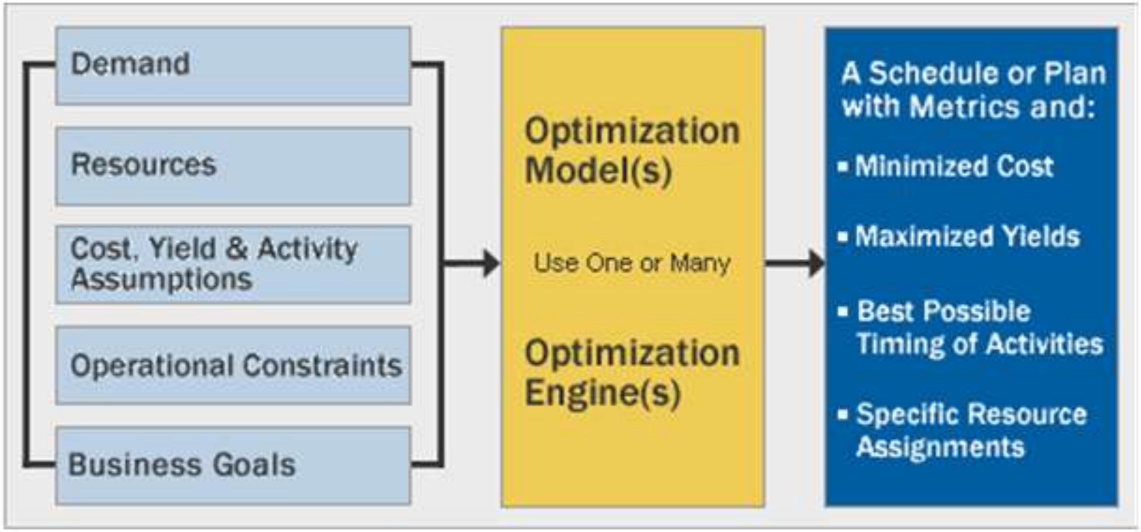
Outline

- **Introduction to Constraint Programming (CP)**
- **JSR-331: oncoming Java CP API standard**
  - allow a user to switch between different CP Solvers without changing a line in the application code
- **Examples of practical use of Constraint Programming for Java-based decision support applications**
  - Demonstrate how CP gives Java developers unprecedented power to define and solve complex constraint satisfaction and optimization problems
  - Integration of Constraint Solvers with Rule Engines and Machine Learning tools

Introduction to CP

- **Constraint Programming (CP) is a very powerful problem solving paradigm with strong roots in Operation Research and AI:**
  - *Handbook of Constraint Programming* (Elsevier, 2006)
  - Association for CP - http://slash.math.unipd.it/acp/
  - Cork Constraint Computation Centre - http://www.4c.ucc.ie/
- **CP is a proven optimization technology introduced to the business application development at the beginning of 1990s**
- **During the 90s ILOG Solver became the most popular optimization tool that was widely used by commercial C++ developers. Being implemented not as a specialized language but rather as an API for the mainstream language of that time, ILOG Solver successfully built a bridge between the academic and business worlds**
- **Nowadays Optimization technology is quickly coming back to the business application development world as an important component of the Enterprise Decision Management (EDM)**

CP as Optimization Technology (taken from ILOG's website)



- Optimization technology helps organizations make better plans and schedules
- A model captures your complex planning or scheduling problem. Then a mathematical engine applies the model to a scenario find the best possible solution
- When optimization models are embedded in applications, planners and operations managers can perform what-if analysis, and compare scenarios
- Equipped with intelligent alternatives, you make better decisions, dramatically improving operational efficiency

Examples of Constraints

- **Constraints represent conditions which restrict our freedom of decision making:**
  - The meeting must start no later than 3:30PM
  - Glass components cannot be placed in the same bin with copper components
  - The job requires Joe or Jim but cannot use John
  - Mary prefers not to work on Wednesday
  - The portfolio cannot include more than 15% of technology stocks unless it includes at least 7% of utility stocks

CSP Example: Procedural Implementation in Java

- **There are 3 integers X, Y, Z defined from 0 to 10.
  Constraints: X<Y and X+Y=Z. Find all feasible values of X, Y, and Z**

- **Simple Java Solution:**

```
for(int x=0; x<11; x++)
    for(int y = 0; y<11; y++)
        for(int z=0; z<11; z++)
            if (x < y && z == x+y)
                System.out.println("X="+x+" Y="+y+" Z="+z);
```

- **"Optimized" Java Solution:**

```
for(int x=0; x<11; x++)
    for(int y = x+1; y<11; y++)
        if (x+y < 11)
            System.out.println("X="+x+" Y="+y+" Z="+(x+y));
```

- **What's wrong with this "solution"?**
  - Readability, Extensibility, Performance,...

CSP Example: Implementation with CP

Simple Solution with Java CP API (JSR-331):

```
// Problem Definition
Problem problem = new Problem("XYZ");
Var x = problem.var("X", 0, 10);
Var y = problem.var("Y", 0, 10);
Var z = problem.var("Z", 0, 10);
x.lt(y).post();  // X < Y
x.add(y).eq(z).post();  // X + Y = Z

// Problem Resolution
Solution[] solutions = problem.getSolver().findAllSolutions();
for(Solution solution : solutions)
      solution.log();
```

How the constraint “X < Y” works

- **Let’s assume X and Y are defined on the domain [0,10]**
- **Initial constraint propagation after posting X<Y constraint:**

  X[0;9]

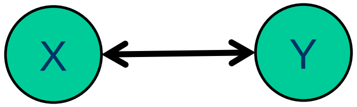
  Y[1;10]

- **Changes in X cause the changes in Y**

  X>3 => Y > 4

- **Changes in Y cause the changes in X**

  Y<=8 => X<=7



- **Bi-Directional constraint propagation**

Constraint Satisfaction Problem - CSP

- **CP clearly separates "What" from "How"**
- **Problem Definition (WHAT):**
  - Constrained Variables with all possible values
    - Integer, Boolean, Real, and Set variables
  - Constraints on the variables
    - Basic arithmetic and logical constraints and expressions
    - Global constraints (AllDifferent, Cardinality, ElementAt, …)
- **Problem Resolution (HOW):**
  - Find Solution(s) that defines a value for each variable such that all constraints are satisfied
    - Find a feasible solution
    - Find an optimal solution
    - Find (iterate through) all solutions
  - Search Strategies

Constraint Satisfaction Environment

- **Predefined classes for Constrained Variables, Constraints, and Search Strategies**
- **Domain representations for major constrained objects**
- **Generic reversible environment**
  - “Try/Fail/Backtrack” capabilities
  - Powerful customizable event management mechanism
  - Constraints use events to control states of all constrained objects
- **Constraint propagation mechanisms**
- **Ability to create problem-specific constraints and search strategies**

Constraint Programming: a bridge between academia and biz

- **CP is especially successful dealing with real-world scheduling, resource allocation, and complex configuration problems:**
  - CP clearly separates problem definition from problem resolution bringing declarative programming to the real-world
  - CP made different optimization techniques handily available to normal software developers (without PhDs in Operation Research)
- **A few real world CP application examples from my consulting practice:**
  - Financial Portfolio Balancing for a Wall Street Wealth Management System
  - Grain Train Scheduling for a Canadian R/R company
  - Truck Loading and Routing system
  - Data Centre Capacity Management
  - Workforce/Workload Scheduling system for a Utility company

Real-world example: Workforce/Workload Management

- **Field Service Scheduling for the Long Island Gas and Electric Utility**
  - More than 1 million customers in Long Island, NY
  - More than 5000 employees
  - Service territory 1,230 square miles
  - Hundreds jobs per day
  - Job requires a mix of people skills, vehicles and equipment
- **Multi-objective Work Planning and Scheduling:**
  - Travel time minimization
  - Resource load levelization
  - Skill utilization (use the least costly skills/equipment)
  - Schedule jobs ASAP
  - Honor user-defined preferences

Some Popular CP Tools

- **CP Modeling Languages**
  - **ILOG OPL** from IBM ILOG (www.ilog.com)
  - **MiniZinc** from G12 group, Australia (http://www.g12.cs.mu.oz.au)
  - **Comet**, Brown University (www.dynadec.com)
  - **Prolog** (ECLiPSe, SICStus)
- **C++ API**
  - **ILOG CP** – Commercial from IBM ILOG
  - **Gecode** – Open Source (www.gecode.org)
- **Java API**
  - **Choco** - Open Source
  - **ILOG JSolver** – Commercial
  - **Constrainer** - Open Source
- **20+ other CP Solvers:** http://slash.math.unipd.it/cp/
- **CP Solvers are usually well integrated with other optimization tools (LP, MIP)**

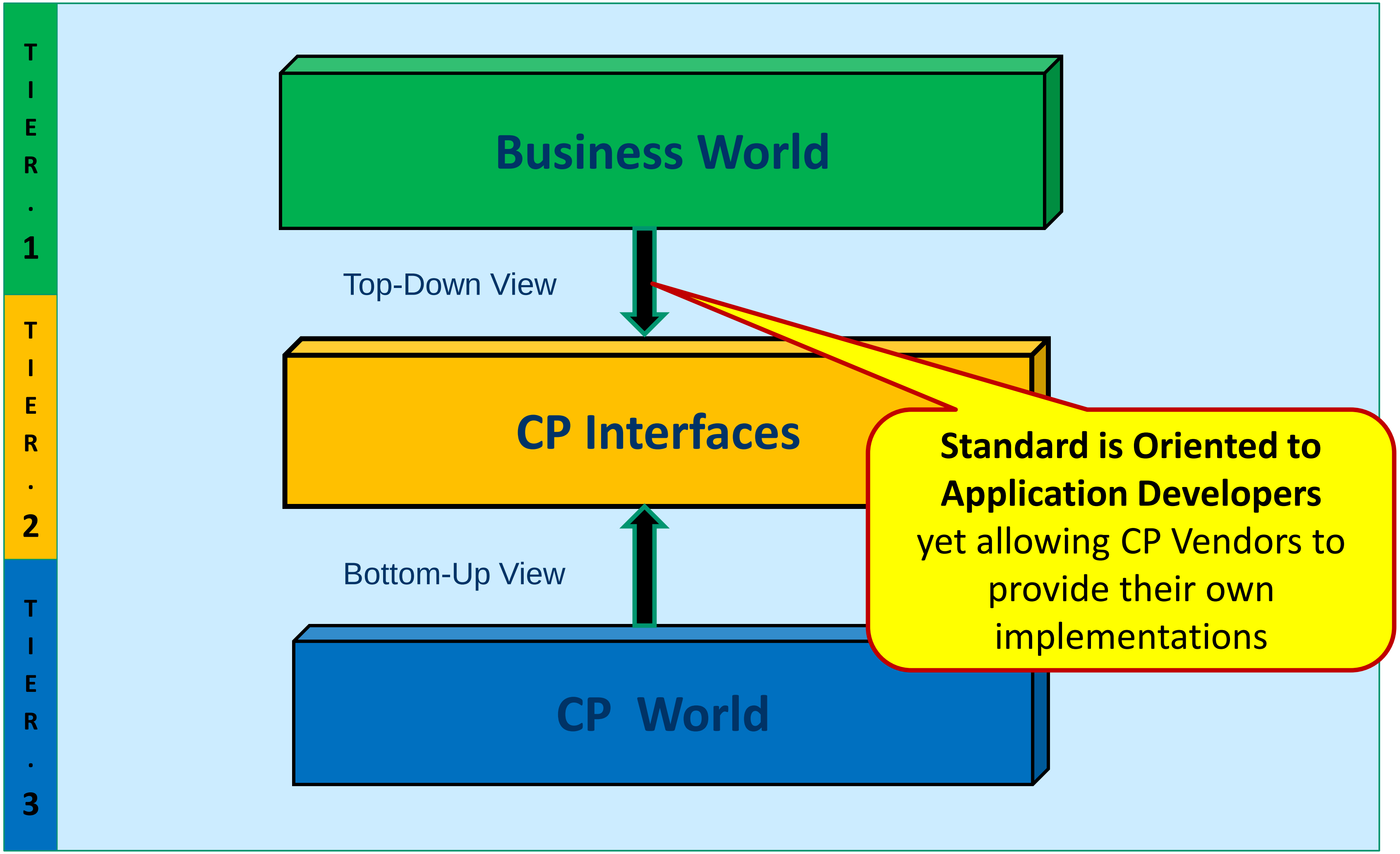
JSR-331 – Java Specification Request

- JSR-331 - Java Constraint Programming API under the roof of the Java Community Process www.jcp.org
- JSR-331 covers key concepts and design decisions related to the standard representation and resolution of constraint satisfaction and optimization problems
- JSR-331 Early Draft is now available for public review www.cpstandards.org

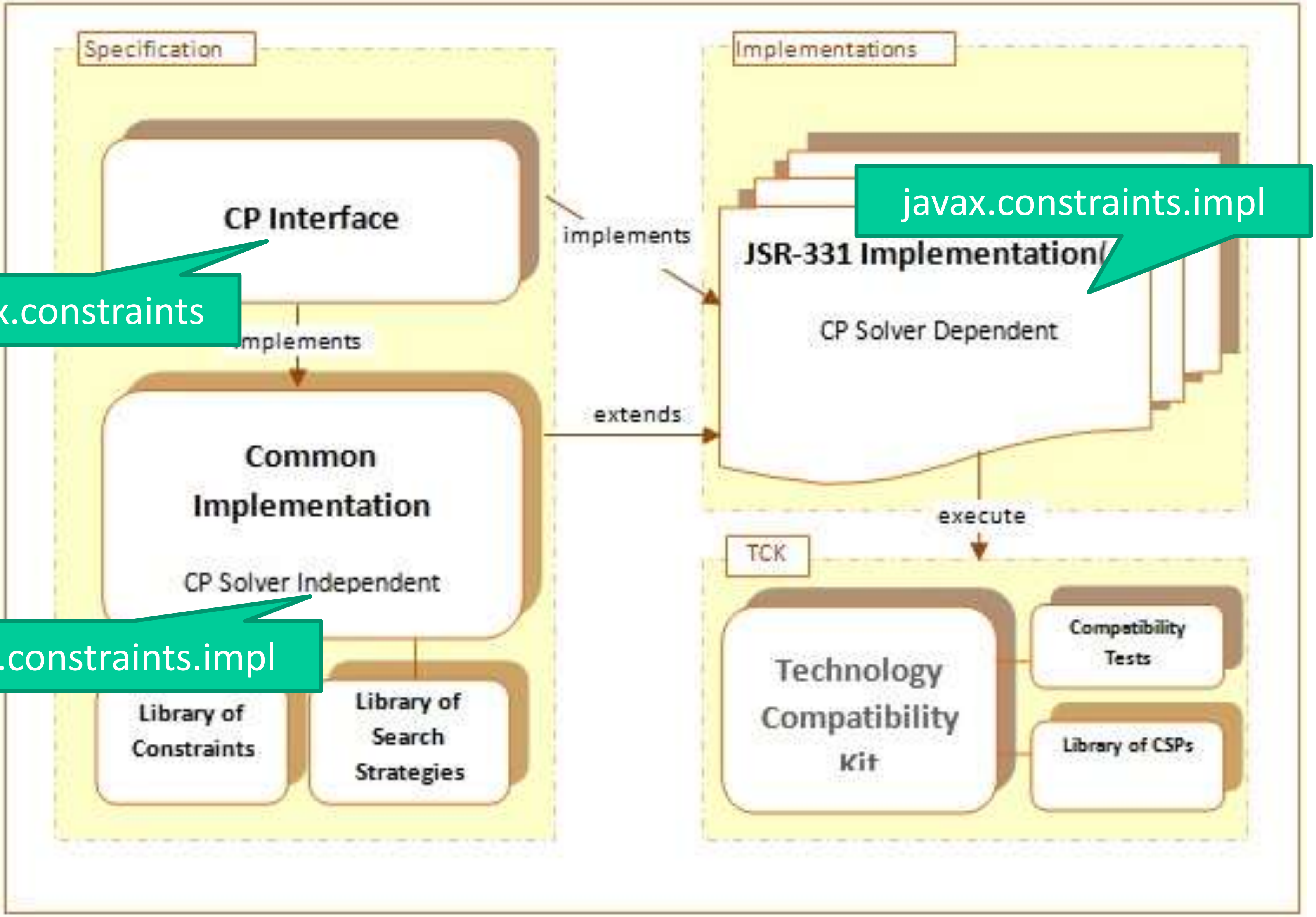
Key Standardization Objectives

- Make Constraint Programming more accessible for business software developers
- Allow a Java business application developer to easily switch between different solver implementations <u>without any(!) changes</u> in the application code
- Assist CP vendors in creating practical JSR-331 implementations

CP Standardization Perspective

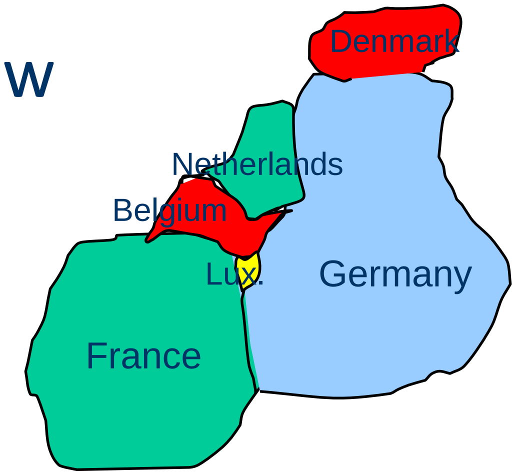


JSR-331 Architecture



CSP Example: "Map Coloring"

- A map-coloring problem involves choosing colors for the countries on a map in such a way that at most 4 colors are used and no two neighboring countries have the same color
- We will consider six countries: Belgium, Denmark, France, Germany, Netherlands, and Luxembourg
- The colors are red, green, blue, yellow



Example "Map Coloring": problem variables

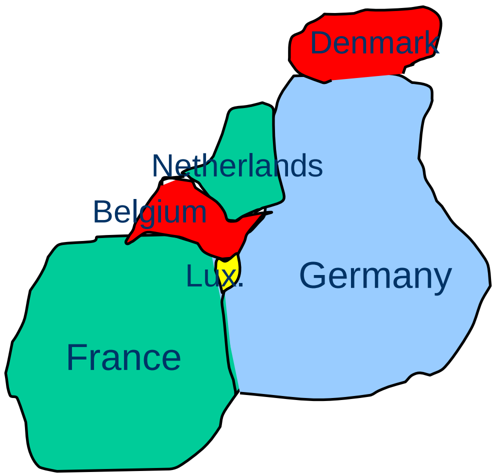
static final String[] *colors = { "red", "green", "blue", "yellow" };*

```
Problem  p = new Problem("Map-coloring");
// Define Variables
Var Belgium       = p.var("Belgium“,0, 3);
Var Denmark       = p.var("Denmark“,0, 3);
Var France        = p.var("France“,0, 3);
Var Germany       = p.var("Germany“,0, 3);
Var Netherlands = p.var("Netherlands“,0, 3);
Var Luxemburg   = p.var("Luxemburg“,0, 3);
```

Each country is represented as a variable that corresponds to an unknown color: 0,1,2, or 3

“Map Coloring”: problem constraints

```
// Define Constraints
France.neq(Belgium).post();
France.neq(Luxemburg).post();
France.neq(Germany).post();
Luxemburg.neq(Germany).post();
Luxemburg.neq(Belgium).post();
Belgium.neq(Netherlands).post();
Belgium.neq(Germany).post();
Germany.neq(Netherlands).post();
Germany.neq(Denmark).post();
```



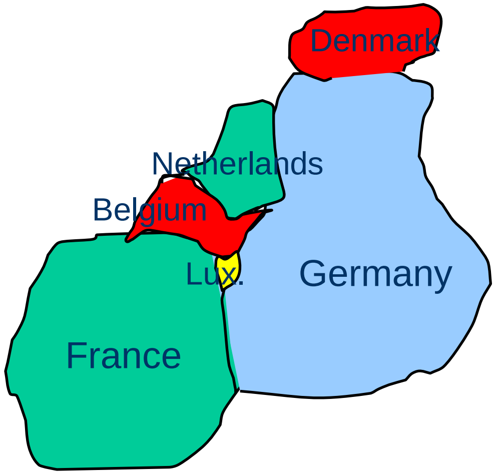

```
// We actually create a constraint and then post it
Constraint c = Germany.neq(Denmark);
c.post();
```

"Map Coloring": solution search

```
// Solve
Goal goal = p.goalGenerate();
Solution solution = p.getSolved().findSolution();
if (solution != null)  {
   for (int i = 0;  i < p.getVars().length;  i++)  {
         Var var = p.getVars()[i];
         p.log(var.getName() + " - " + colors[var.getValue()]);
   }
}
```

// Solution:

Belgium – red
Denmark – red
France – green
Germany – blue
Netherlands – green
Luxemburg - yellow



Over-Constrained Problems

- **In real-world many problems are over-constrained. If this is a case, we may want to find a solution that minimizes the total constraint violation**
- **Consider a map coloring problem when there are no enough colors, e.g. only three colors:**
  - Coloring violations may have different importance for France – Belgium and France – Germany
  - Find a solution that minimizes total constraint violations

23

Over-Constrained Map Coloring Problem

- **Constraint “softening” rules:**

  Coloring constraint violations have different importance on the scale 0-9999:

  Luxemburg– Germany (9043)

  France – Luxemburg (257)

  Luxemburg – Belgium (568)

- **We want to find a solution that minimizes the total constraint violation**

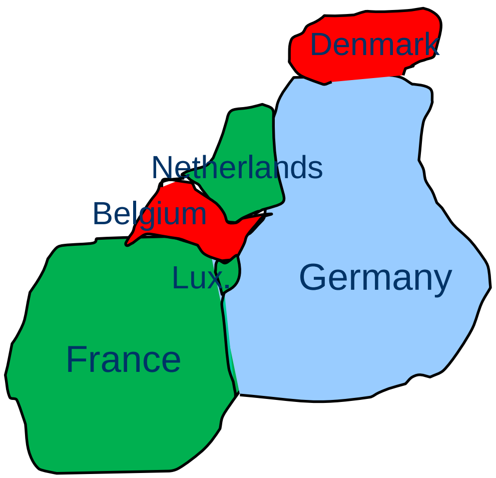
Map Coloring with Hard and Soft Constraints

```
// Hard Constraints
France.neq(Belgium).post();
France.neq(Germany).post();
Belgium.neq(Netherlands).post();
Belgium.neq(Germany).post();
Germany.neq(Denmark).post();
Germany.neq(Netherlands).post();
// Soft Constraints
Var[] weightVars = {
    Luxemburg.eq(Germany).asBool().mul(9043),
    France.eq(Luxemburg).asBool().mul(257),
    Luxemburg.eq(Belgium).asBool().mul(568)
};
Var weightedSum = p.sum(weightVars);
```

Luxemburg – Germany (9043)
France – Luxemburg (257)
Luxemburg – Belgium (568)

Minimize Total Constraint Violations

```
// Optimal Solution Search

Solution solution =  p.getSolver().getOptimalSolution(weightedSum);

if (solution == null)

        p.log("No solutions found");

else

        solution.log();
```

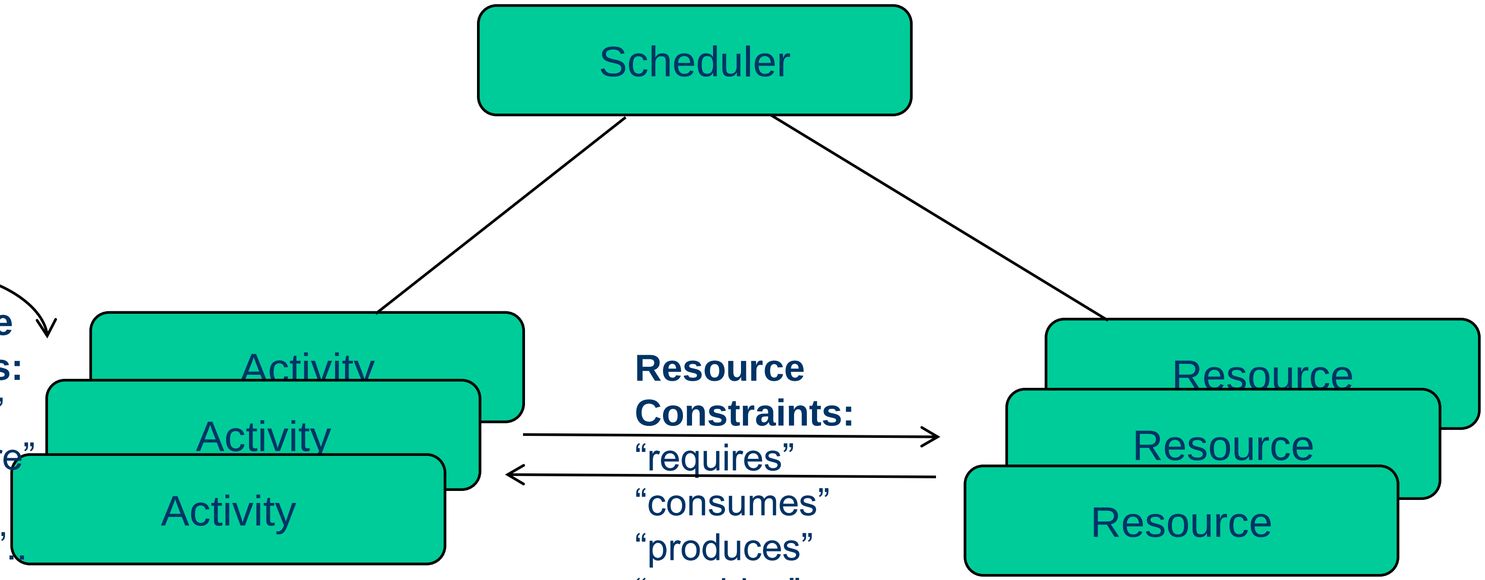


Solution:
Belgium[0] Denmark[0] France[1] Germany[2] Netherlands[1] Luxemburg[1]

26

- **Scheduling problems usually deals with:**
  - Activities with yet unknown start times and known durations (not always)
  - Resources with limited capacities varying over time
  - Constraints:
    - Between activities (e.g. Job2 starts after the end of Job1)
    - Between activities and resources (e.g. Job1 requires a welder, where Jim and Joe both have a welder skills)
- **There are multiple scheduling objectives (e.g. minimize the makespan, utilize resources, etc.)**

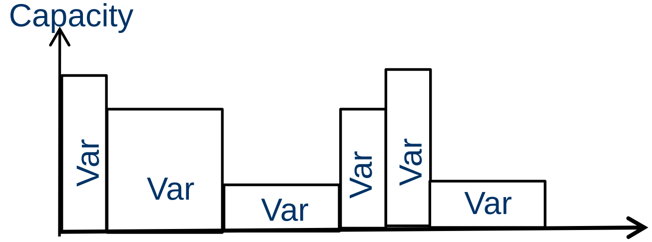
How we may create a CP-based Scheduler?



Precedence Constraints:
“starts after”
“starts before”
“starts at”
“and before”…

- Var start
- Var duration
- Var end

Resource Constraints:
“requires”
“consumes”
“produces”
“provides”

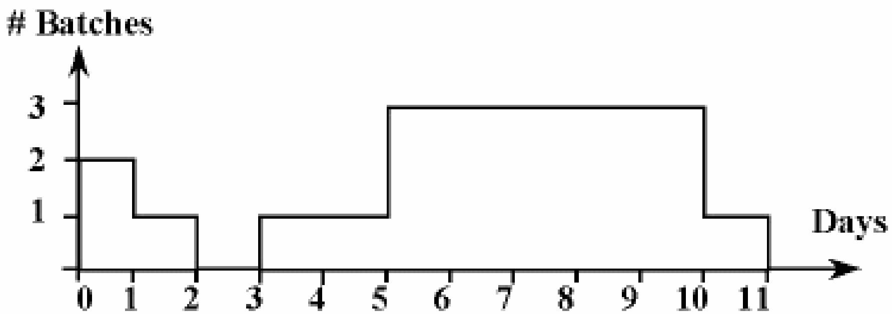


Capacity Timetable

```
// Alternative resource requirements
activity1.requires(resource2, varReq2).post();
activity1.requires(resource3, varReq3).post();
varReq2.ne(varReq3).post();
```

Oven - job scheduling with one resource

There is an oven in which we can fire batches of bricks. There are five orders to fire X batches during Y days. Schedule all orders to be done in no more than 11 days taking into consideration the following oven availability:



Global capacity of the oven

A 2 batches, 1 day

B 1 batch, 4 days

C 1 batch, 4 days

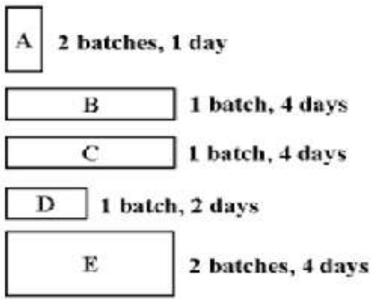
D 1 batch, 2 days

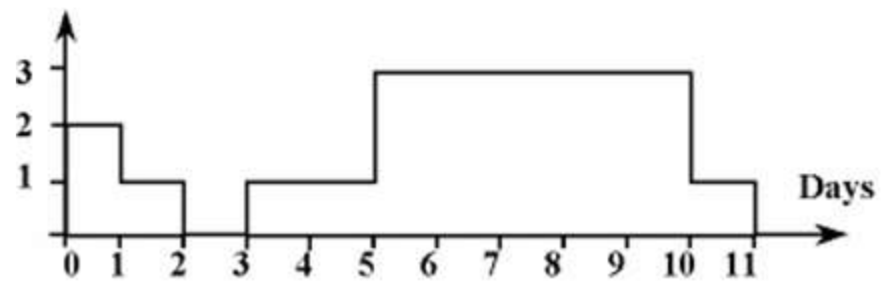
E 2 batches, 4 days

5 Activities

Scheduling Sample Implementation

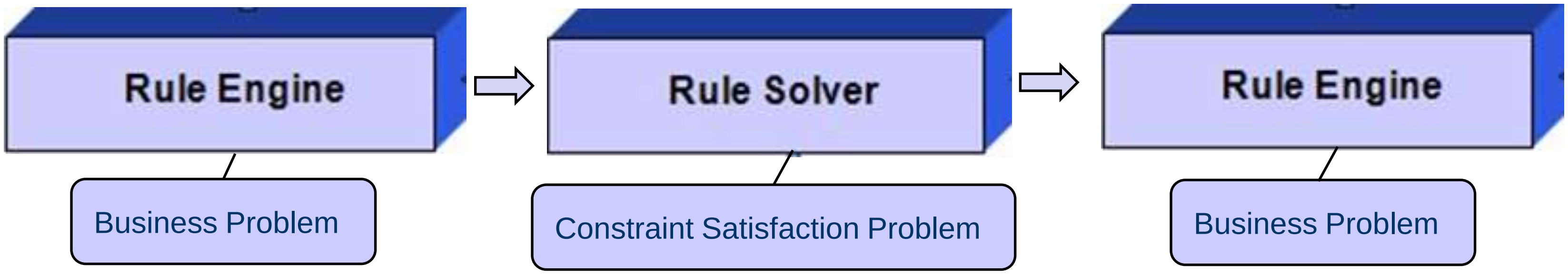

```
Problem problem = new Problem("Oven Scheduling Example");
Schedule schedule = problem.addSchedule(0, 11);
Activity A = schedule.addActivity(1, "A");
Activity B = schedule.addActivity(4, "B");
Activity C = schedule.addActivity(4, "C");
Activity D = schedule.addActivity(2, "D");
Activity E = schedule.addActivity(4, "E");
Resource oven = schedule.addResource(3, "oven");
oven.setCapacityMax(0, 2);
oven.setCapacityMax(1, 1);
oven.setCapacityMax(2, 0);
oven.setCapacityMax(3, 1);
oven.setCapacityMax(4, 1);
oven.setCapacityMax(10, 1);
// Resource Constraints
A.requires(oven, 2).post();
B.requires(oven, 1).post();
C.requires(oven, 1).post();
D.requires(oven, 1).post();
E.requires(oven, 2).post();
// Find Solution
schedule.scheduleActivities();
schedule.displayActivities();
```



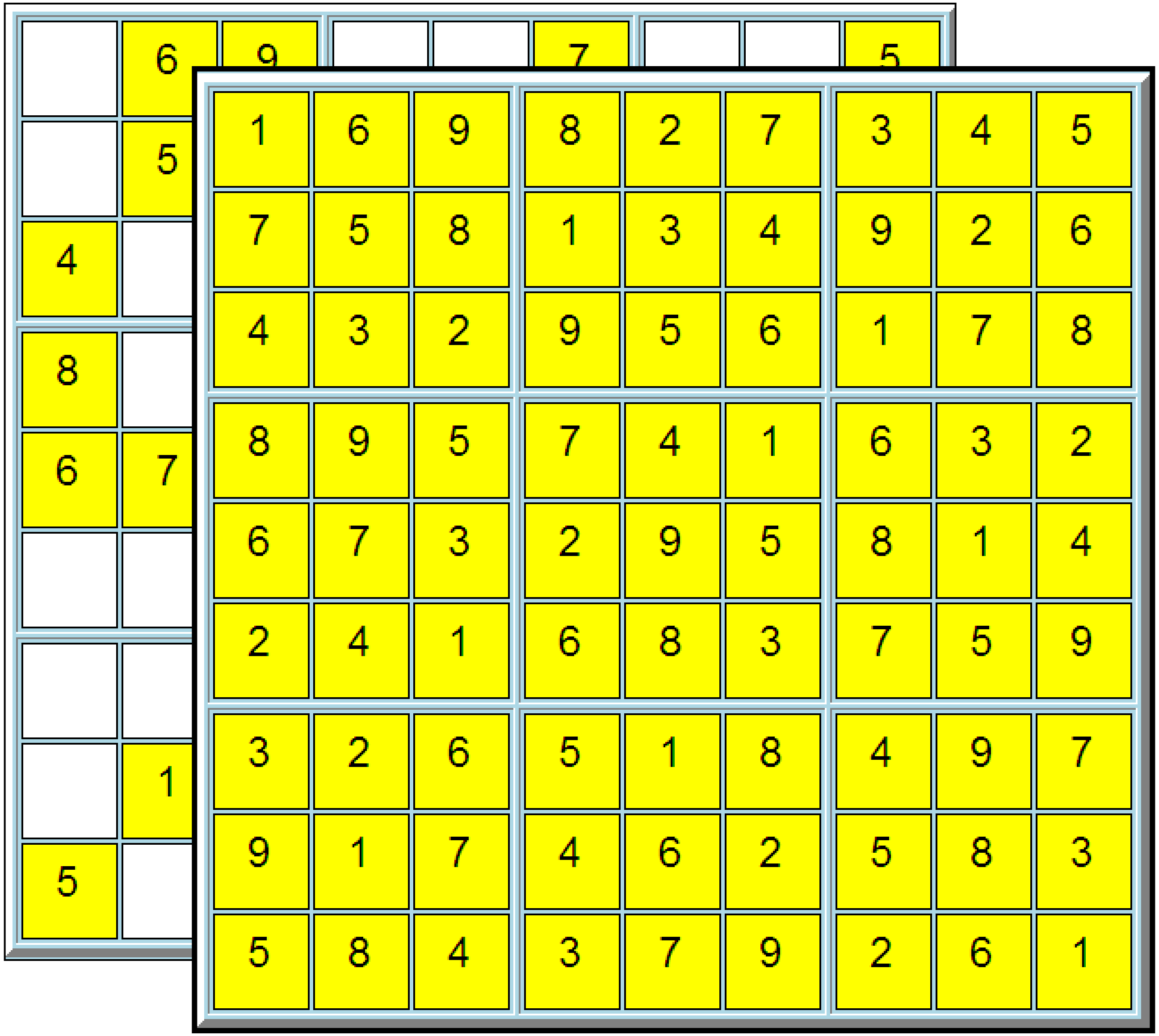


SOLUTION:
A[5 -- 1 --> 6) requires oven[2]
B[3 -- 4 --> 7) requires oven[1]
C[7 -- 4 --> 11) requires oven[1]
D[0 -- 2 --> 2) requires oven[1]
E[6 -- 4 --> 10) requires oven[2]

- Business Rules could be used to define and modify business objects
- Rule Engine can generate a related constraint satisfaction problem/subproblem representing it in terms of constrained variables and constraints
- CP Solver can solve the optimization problems and return the results to the Rules Engine for further analysis

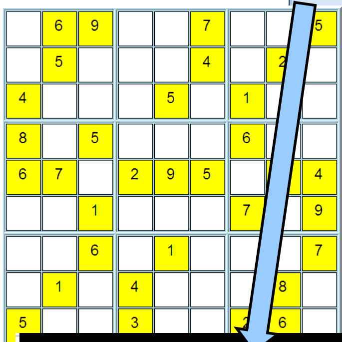


Notorious CSP “SUDOKU”



1	6	9	8	2	7	3	4	5
7	5	8	1	3	4	9	2	6
4	3	2	9	5	6	1	7	8
8	9	5	7	4	1	6	3	2
6	7	3	2	9	5	8	1	4
2	4	1	6	8	3	7	5	9
3	2	6	5	1	8	4	9	7
9	1	7	4	6	2	5	8	3
5	8	4	3	7	9	2	6	1

Sudoku Constraints in OpenRules Excel Rules Table



Rules void postSudokuConstraints(CpProblem p)									
Array Name	**Variables**								
row0	x00	x01	x02	x03	x04	x05	x06	x07	x08
row1	x10	x11	x12	x13	x14	x15	x16	x17	x18
row2	x20	x21	x22	x23	x24	x25	x26	x27	x28
row3	x30	x31	x32	x33	x34	x35	x36	x37	x38
row4	x40	x41	x42	x43	x44	x45	x46	x47	x48
row5	x50	x51	x52	x53	x54	x55	x56	x57	x58
row6	x60	x61	x62	x63	x64	x65	x66	x67	x68
row7	x70	x71	x72	x73	x74	x75	x76	x77	x78
row8	x80	x81	x82	x83	x84	x85	x86	x87	x88

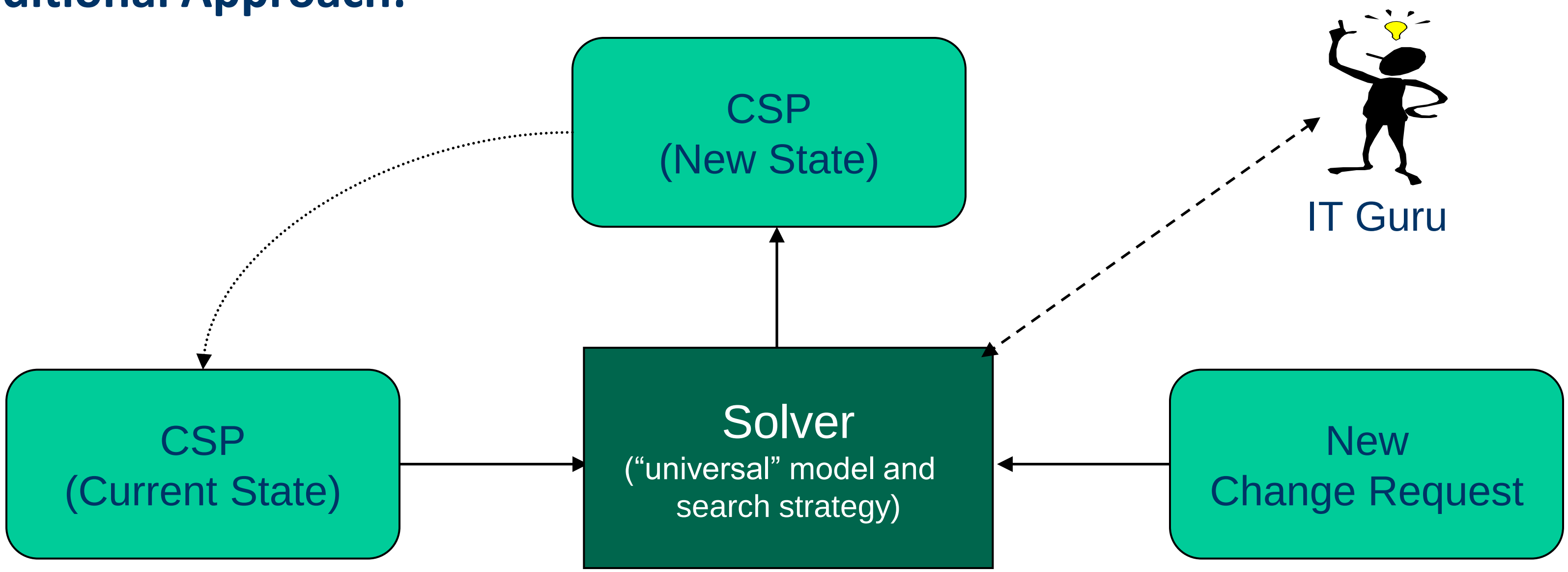
Row Constraints

Rules void postSudokuConstraints(CpProblem p)	
Action	
CpVariable[] array = p.addArray(name,vars); p.**allDiff**(array).post();	
String name	String[] vars
Array Name	**Variables**

block00	x00	x01	x02	x10	x11	x12	x20	x21	x22
block01	x03	x04	x05	x13	x14	x15	x23	x24	x25
block02	x06	x07	x08	x16	x17	x18	x26	x27	x28
block10	x30	x31	x32	x40	x41	x42	x50	x51	x52
block11	x33	x34	x35	x43	x44	x45	x53	x54	x55
block12	x36	x37	x38	x46	x47	x48	x56	x57	x58
block20	x60	x61	x62	x70	x71	x72	x80	x81	x82
block21	x63	x64	x65	x73	x74	x75	x83	x84	x85
block22	x66	x67	x68	x76	x77	x78	x86	x87	x88

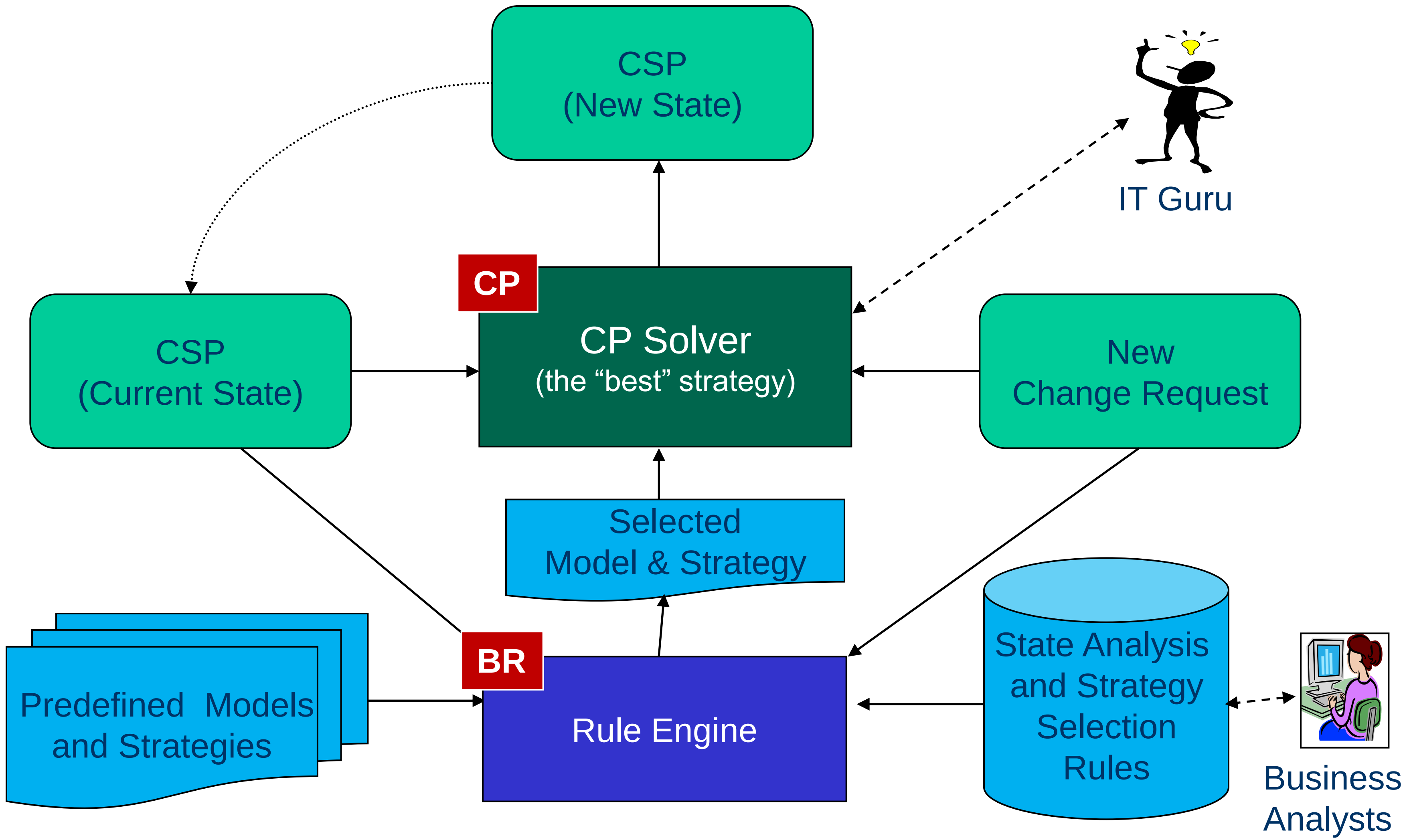
Block Constraints

- **Typical Online Systems with CP-based Solvers:**
  - Online Reservation systems (hotels, tours, vacations, ..)
  - Event Scheduling (both business and personal events in social networks)
  - Field Service Scheduling, Advertisement Scheduling, and more
- **Traditional Approach:**

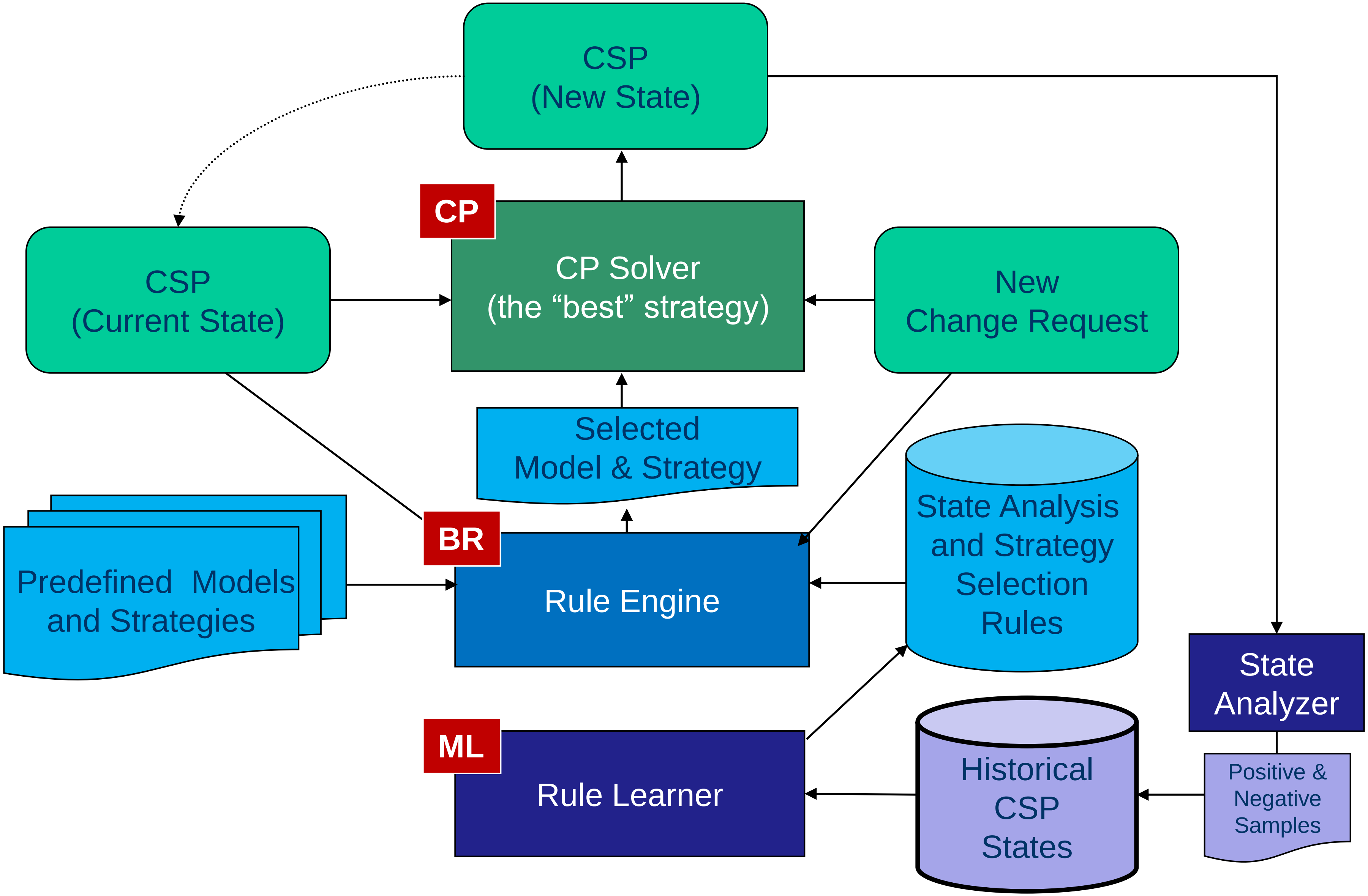


  - "Fat" Problem Solver tuned for all possible problem states
  - Complexity grows over time – hard to create and maintain

Online Decision Support: CP + BR
adding Rule Engine to find the "best" strategy



Online Decision Support: CP + BR + ML adding Rule Learner to find the “best” strategy



Summary

- **Constraint Programming empowers application developers with sophisticated decision-support (optimization) capabilities**
- **Proven CP + BR methodology and supporting open source and commercial tools are available in a vendor-neutral way (JSR-331)**
- **Online decision support may be done with**
  - **CP or BR** only: Hard to create and maintain "fat" Solvers controlled by IT
  - **CP + BR**: Rule Engine recommends a CSP model and search strategy based on business rules controlled by business analysts
  - **CP + BR + ML**: Rule Learner discovers model/strategy selection rules based on historical Solver runs – **"Ever-learning" decision support**!

Q & A jacobfeldman@openrules.com

epicenter
The Irish Software Show 2010

Aol
Microsoft Visual Studio
IBM
irish dev
dediserve
slattery communications
qms
qms-software.com
unum
Better benefits at work.
DNM TECHNOLOGY
POWERING YOUR POTENTIAL
neon sms
Cincom
software SKILLNET
learn, share, grow
eset
cloud ICT
Brightwater
recruitment specialists
Cinfinity
Digital Mines
Utility Computing
sagepay
sonru.com
Pop Cap
WIBU SYSTEMS
clarify
communications consulting ltd
black marble