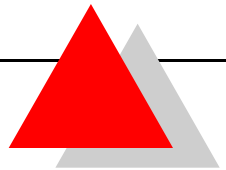
Finite Domain Constraint Programming Methodology

Helmut Simonis
COSYTEC SA

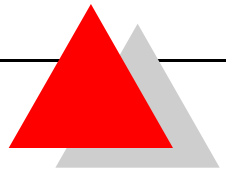
Outline

- **Introduction**
- **Modeling**
  - **Objects**
  - **GUI**
  - **Constraint**
  - **Search**
- **Examples**
  - **Ship loading**
  - **Chemical Process**
  - **Two stage process**
- **Golden rules**



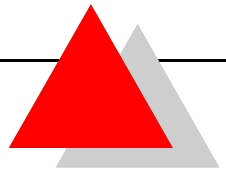
Motivation

- **Only 16 % of all software projects finish on time and on budget**
  - **Fast Magazine, 1996**
- **The average software project is 6-12 month late and 50-100 percent over budget**
  - **E. Yourdon: Death March, Prentice Hall, Upper Saddle River, NJ, 1997**



Background

- **CHIC ESPRIT Project**
  - **A. Chamard, B. Fischer, B. Guinaudeau, A. Guilland: CHIC Lessons on CLP Methodology**
  - **www.ecrc.de/eclipse/html/CHIC_Methodology.html**
- **CHIC-2 ESPRIT Project**
  - **C. Gervet: CHIC2 Methodology**
  - **www-icparc.doc.ic.ac.uk/chic2/chic2_methodology/index.html**
- **B. Smith, Leeds Univ**
  - **The Art of Modelling, ConsNet Meeting, Edinbourgh, Sept 1999**
- **H. Simonis**
  - **A Problem Classification Scheme, PACT 97, London April 1997**

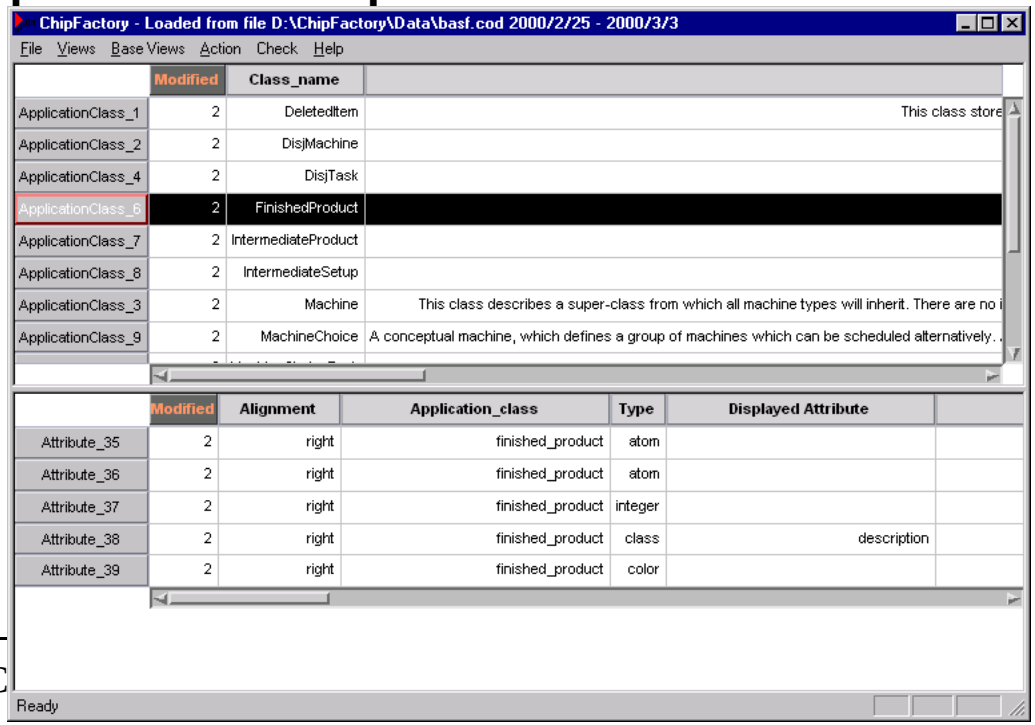


Objectives

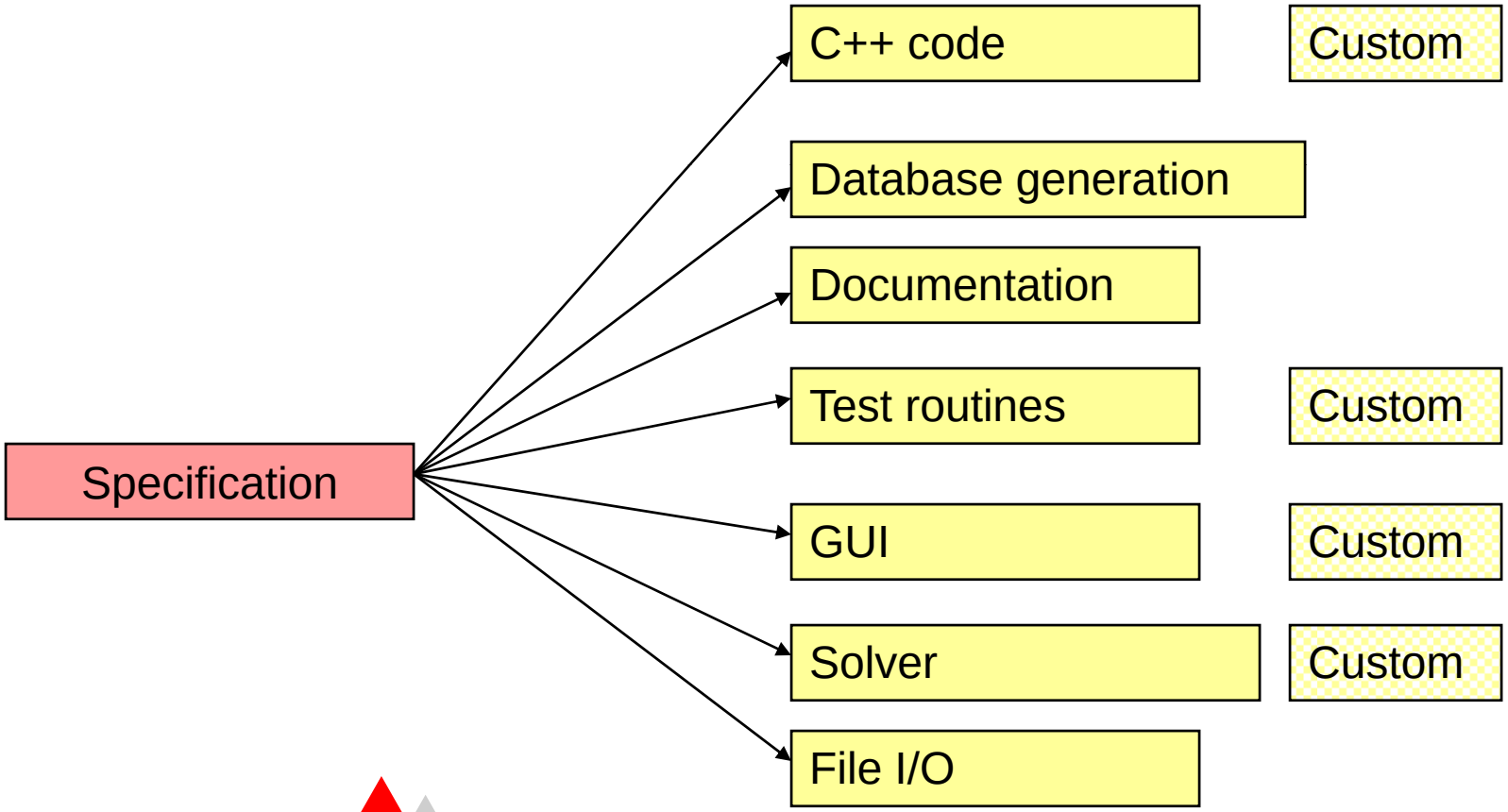
- **Fast development**
- **Consistent design**
- **Flexible program**
- **Complete GUI**
- **Data management**

Approach

- **CHIP Factory**
- **Program generator for C++**
- **CHIP constraint library**
- **Generates (complete) application from specification**

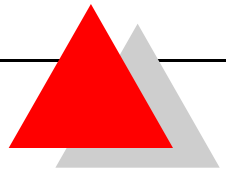


Principles of Operation



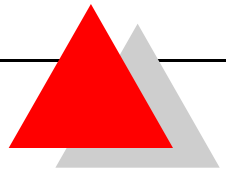
Modeling

- **Convert ideas into program**
- **Rapid prototyping/development**
- **Iterative process**



Application Parts to Model

- **Object model**
  - **GUI design**
  - **database model**
- **Constraint model**
  - **Search procedure**

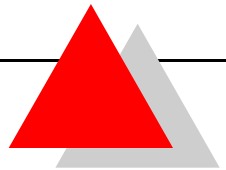


Part I

Modeling

Object Model

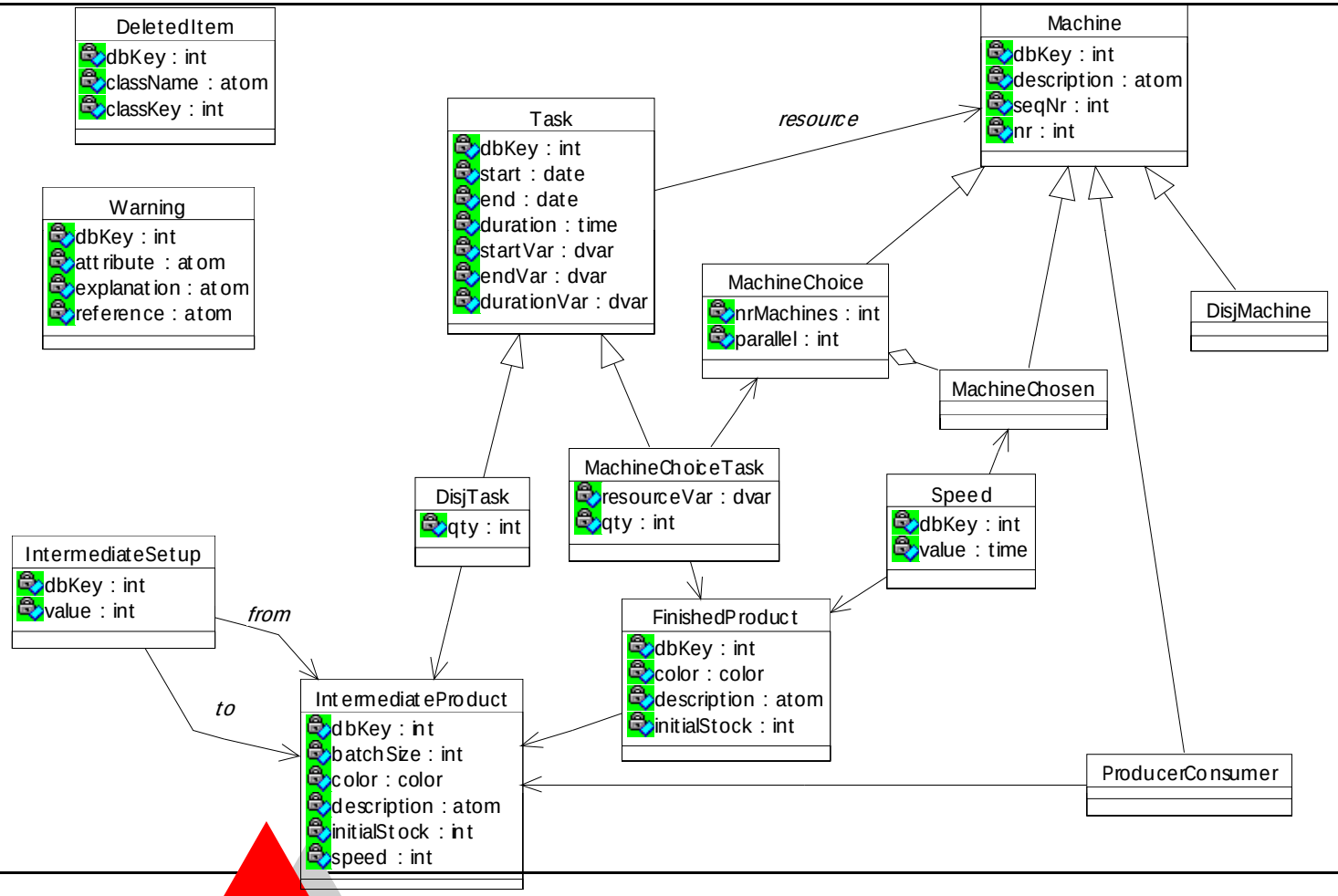
- **Use subset of UML**
- **Important concepts**
  - **Classes**
  - **Attributes**
  - **Predefined data types**
  - **Instances**
  - **Relations**



UML

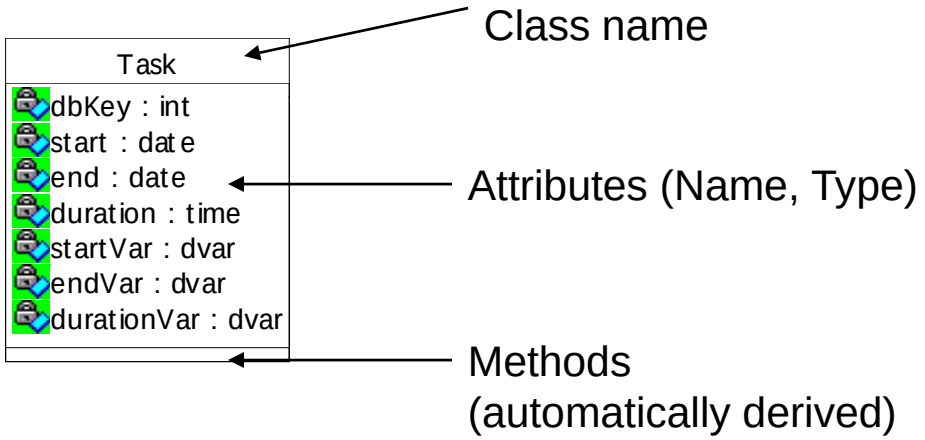
- **Unified Modeling Language**
  - **defined 1995**
  - **continued development**
  - **“graphical language for specifying artifacts of a software intensive system”**
- **Designed by**
  - **G. Booch, I. Jacobson, J. Rumbaugh et al**
- **Basis of many software engineering tools**
- **Links**
  - **www.omg.org**
  - **www.rational.com**
  - **series of books from Addison Wesley**
- **Does many things we don’t need here**

Example Model



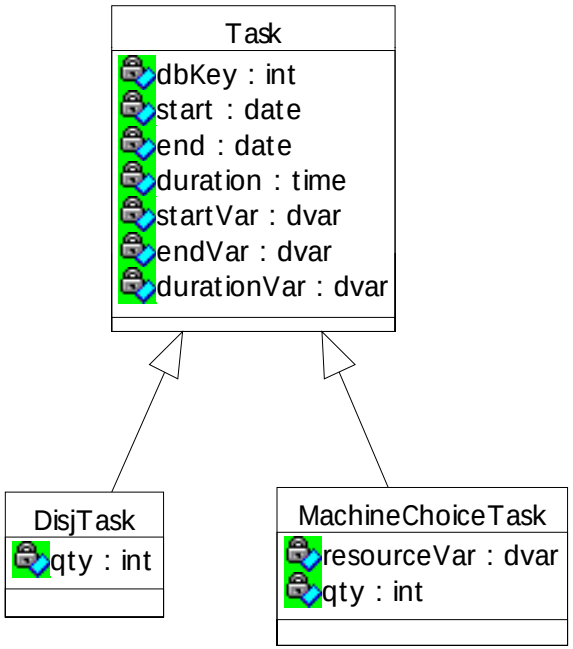
Classes

- **Described by class name, attributes and links**



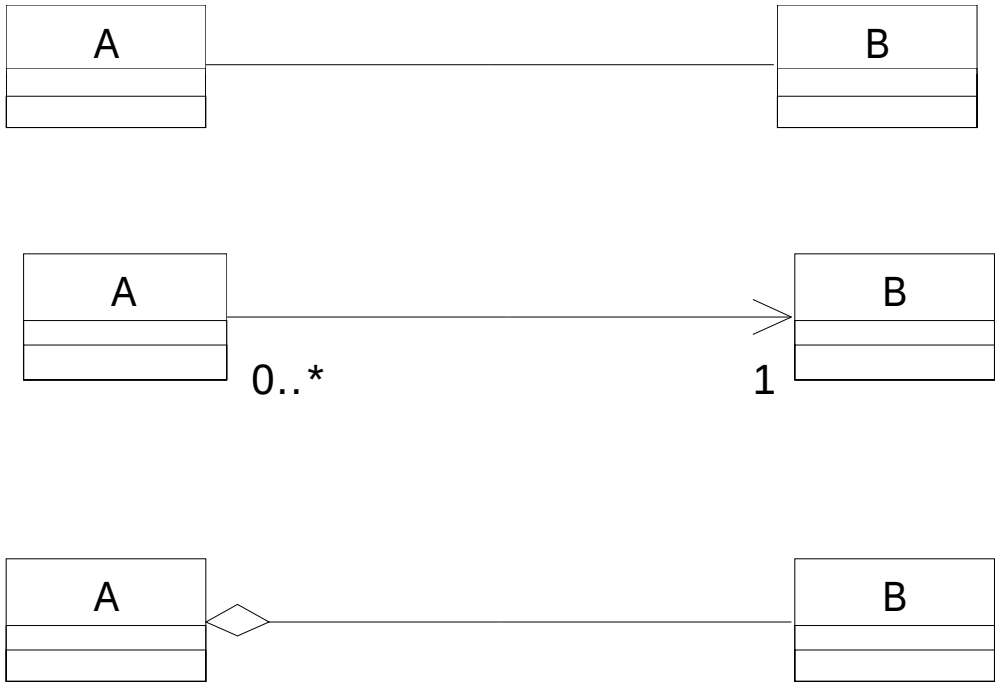
Derivation

- **New classes can be derived from existing ones**



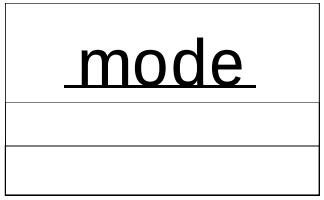
Associations, Navigation, Aggregation

- **Links between classes, which allow navigation between them**



Instances

- **You can specify particular instances in your model**
- **Usually, instances are created from data and are not represented**

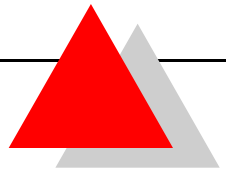


Predefined Data Types

- **integer**
- **bool**
- **double**
- **atom**
- **class**
- **toggle**
- **date**
- **time**
- **color**
- **dvar**

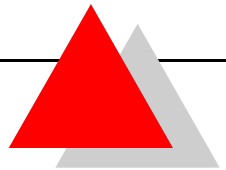
Predefined Classes

- **Machine**
- **Task**
- **Warning**
- **DeletedItem**
- **Solver**
- **GeneralParameters**



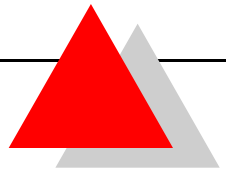
I/O Design

- **Interface via database**
- **Each class corresponds to a table**
  - **class hierarchy ignored**
  - **except for primary key generation**
- **Each attribute corresponds to a table column**
- **Simple file I/O by dumping/restoring objects**



GUI Design

- **Present all data in different views**
- **Data manipulation for all objects in application**
- **Predefined display elements**
  - **list**
  - **double list**
  - **matrix**
  - **gantt**
  - **chart**



List

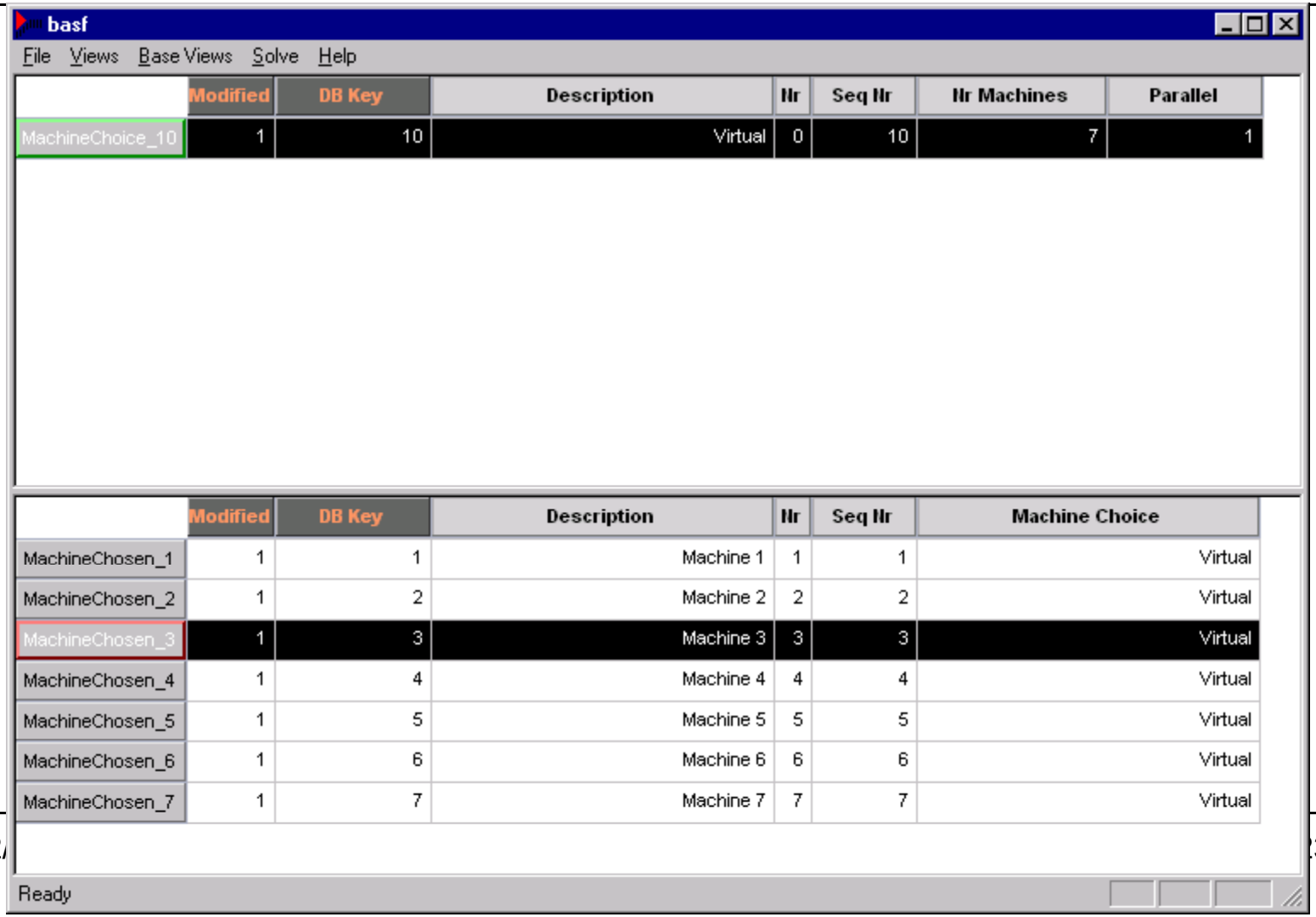
basf

File Views Base Views Solve Help

	Modified	DB Key	Description	Nr	Seq Nr	Intermediate Product
ProducerConsumer_1	1	1	P/C 1	0	1	IP 1
ProducerConsumer_2	1	2	P/C 2	0	1	IP 2
ProducerConsumer_3	1	3	P/C 3	0	1	IP 3
ProducerConsumer_4	1	4	P/C 4	0	1	IP 4
ProducerConsumer_5	1	5	P/C 5	0	1	IP 5
ProducerConsumer_6	1	6	P/C 6	0	1	IP 6

Ready

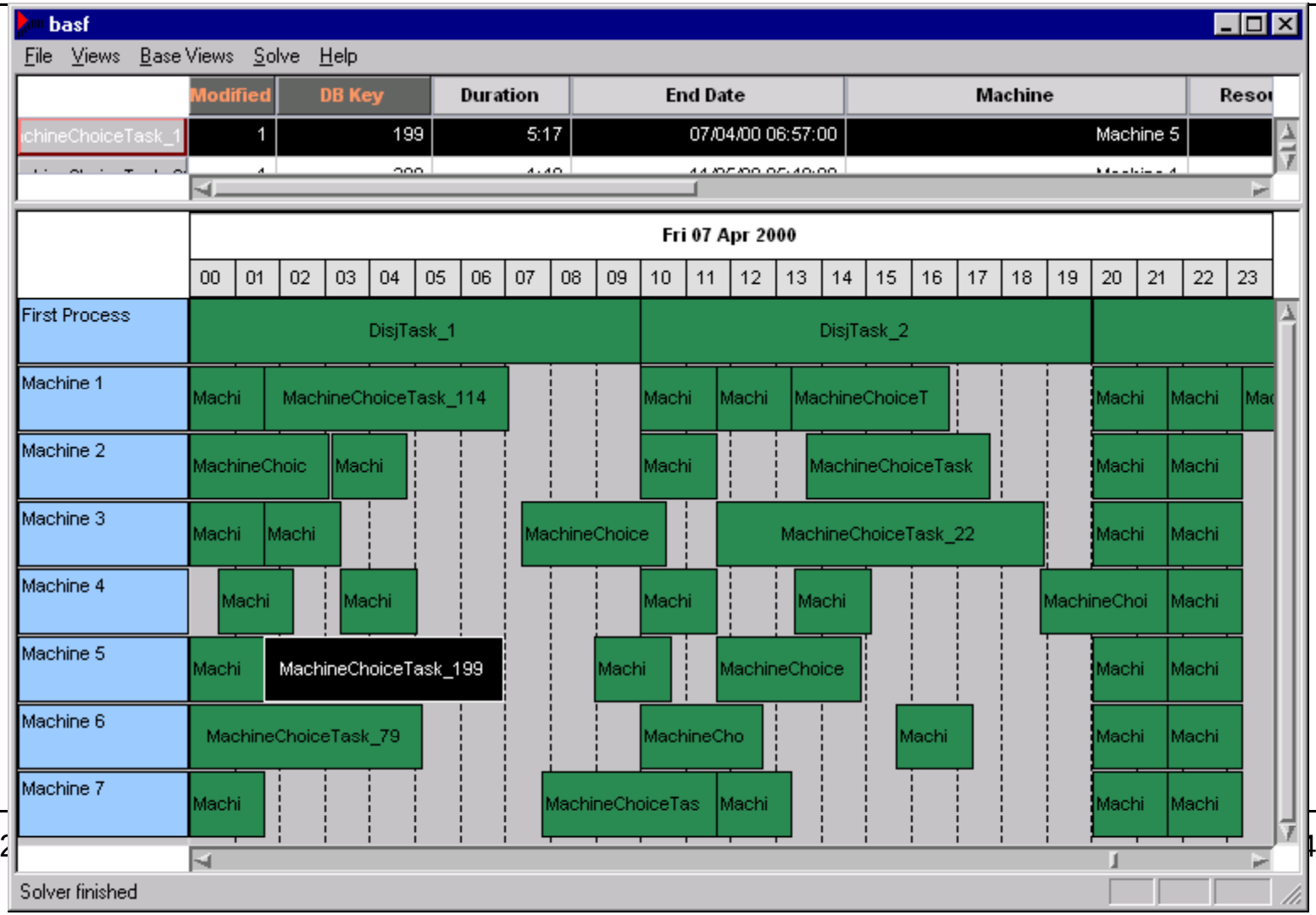
Double List



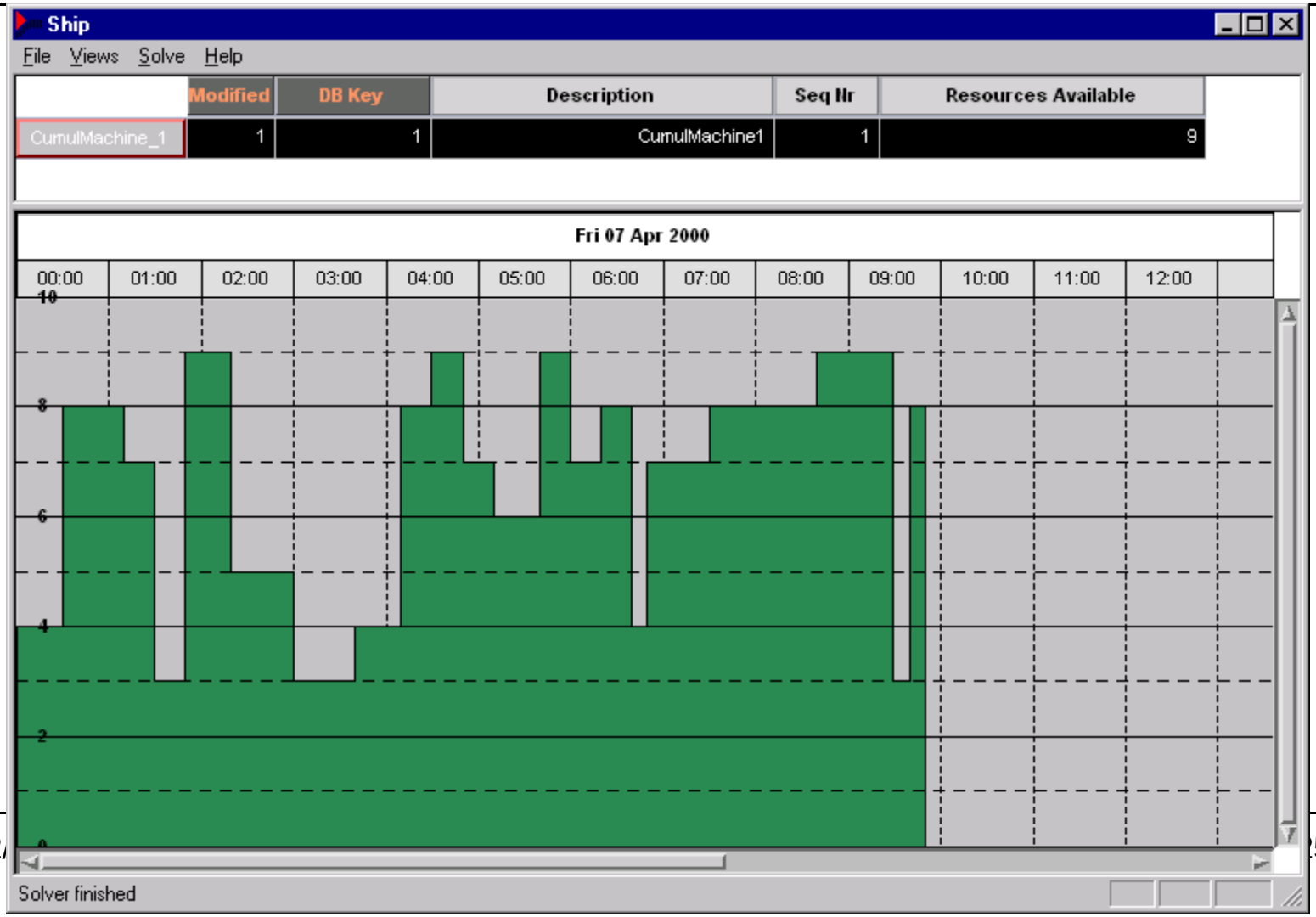
	Modified	DB Key	Description	Nr	Seq Nr	Nr Machines	Parallel
MachineChoice_10	1	10	Virtual	0	10	7	1

	Modified	DB Key	Description	Nr	Seq Nr	Machine Choice
MachineChosen_1	1	1	Machine 1	1	1	Virtual
MachineChosen_2	1	2	Machine 2	2	2	Virtual
MachineChosen_3	1	3	Machine 3	3	3	Virtual
MachineChosen_4	1	4	Machine 4	4	4	Virtual
MachineChosen_5	1	5	Machine 5	5	5	Virtual
MachineChosen_6	1	6	Machine 6	6	6	Virtual
MachineChosen_7	1	7	Machine 7	7	7	Virtual

Gantt


basf
File Views Base Views Solve Help
Modified DB Key Duration End Date Machine
1 199 5:17 07/04/00 06:57:00 Machine 5
Fri 07 Apr 2000
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
First Process
Machine 1
Machine 2
Machine 3
Machine 4
Machine 5
Machine 6
Machine 7
DisjTask_1
DisjTask_2
Machi
MachineChoiceTask_114
MachineChoiceT
MachineChoic
MachineChoiceTask
MachineChoice
MachineChoiceTask_22
MachineChoi
MachineChoiceTask_199
MachineChoice
MachineChoiceTask_79
MachineCho
MachineChoiceTas
Solver finished

Chart

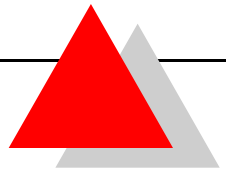

Ship
File Views Solve Help
Modified
DB Key
Description
Seq Nr
Resources Available
CumulMachine_1
1
1
CumulMachine1
1
9
Fri 07 Apr 2000
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
10
8
6
4
2
0
Solver finished

GUI design (2)

- **Interaction**
- **Menus/Menu items**

Constraint Models

- **Solver**
- **Variables**
- **Constraints**
- **Search procedure**



Predefined Solver Class

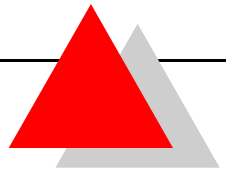
- **Describes overall approach to problem**
- **Basic class from which individual solvers are derived**
- **Possibly more than one solver in an application**
- **Fundamental operations**
  - **initialize**
  - **define variables**
  - **define constraints**
  - **enumerate**
  - **terminate**

Solver Class Methods

- **General description of solver**
- **Cost variable**
- **Time resolution**
- **Search limits (choices, time, interactive)**
- **Error handling**
- **Visualization**
- **Log file**
- **Conversion methods (date,time,class <-> dvar)**

Variables

- **Special attributes of classes**
- **Some solver may use subset of variables**
- **Domain size given for each variable**

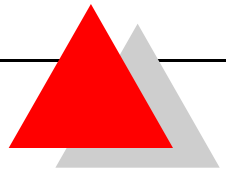


Constraints

- **Special methods of classes**
- **Description in modeling language**
- **Full language defined in LISCOS project**
- **Current design Prolog based**

Modeling Languages

- **Well established in MIP field**
  - **AMPL**
  - **GAMS**
  - **MP Model**
  - **BC-Prod**
- **Relatively new for CP**
  - **OPL (Van Hentenryck)**
  - **PLAM (Bockmayr)**

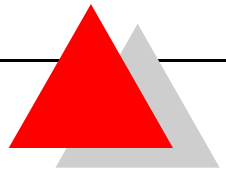


LISCOS

- **5th Framework project**
- **Development of large-scale supply chain management software**
- **Combination of MIP/Polyhedral Cuts/CP solvers**
- **Partners**
  - **BASF**
  - **Dash**
  - **COSYTEC**
  - **CORE**
  - **Proctor and Gamble**
  - **PSA**
  - **Barbot**
  - **LORIA**
  - **DEIO**

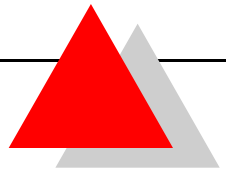
Search Procedure

- **Type of search**
- **Type of optimization**
- **Type of tree traversal**



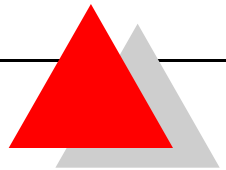
Type of Search

- **Value choice**
  - **select a variable (deterministic)**
  - **choose a value (non-deterministic)**
- **Variable choice**
  - **choose a variable (non-deterministic)**
  - **select a value (deterministic)**
- **Snake type**
  - **choose a value (non-deterministic)**
  - **this selects the next variable (fixed)**



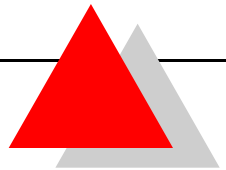
Type of Optimization

- **none**
  - **find one (good) feasible solution**
  - **does not look at alternatives**
- **min_max**
  - **restart search with new cost constraint whenever solution is found**
  - **positive: use upper bound for branching decisions**
  - **negative: may explore part of tree several times**
- **minimize**
  - **continue search re-stating cost constraint after each backtrack**
  - **positive: explore each node at most once**
  - **negative: tree structure fixed initially**



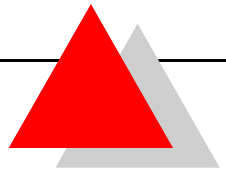
Tree Traversal

- **Complete search**
  - **explore all nodes in tree**
  - **too expensive for most problems**
  - **limits search to small part of tree**
- **Partial search**
  - **LDS**
    - **search around heuristic solution**
  - **credit**
    - **explore top of tree completely**
  - **barrier**
    - **restart LDS at each local failure**



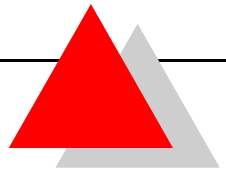
Problems with Search

- **No known a priori selection of methods**
  - **Meta-heuristics factory (Bouygues, CP 99)**
- **Each problem (data set) can lead to new method**
- **Over-tuning of enumeration method to one data-set**



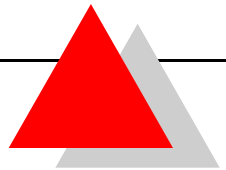
Part II

Examples



Example I

Ship Loading



Problem Description

- **Schedule unloading/loading operations for a ship**
- **Set of operations with different duration and manpower use**

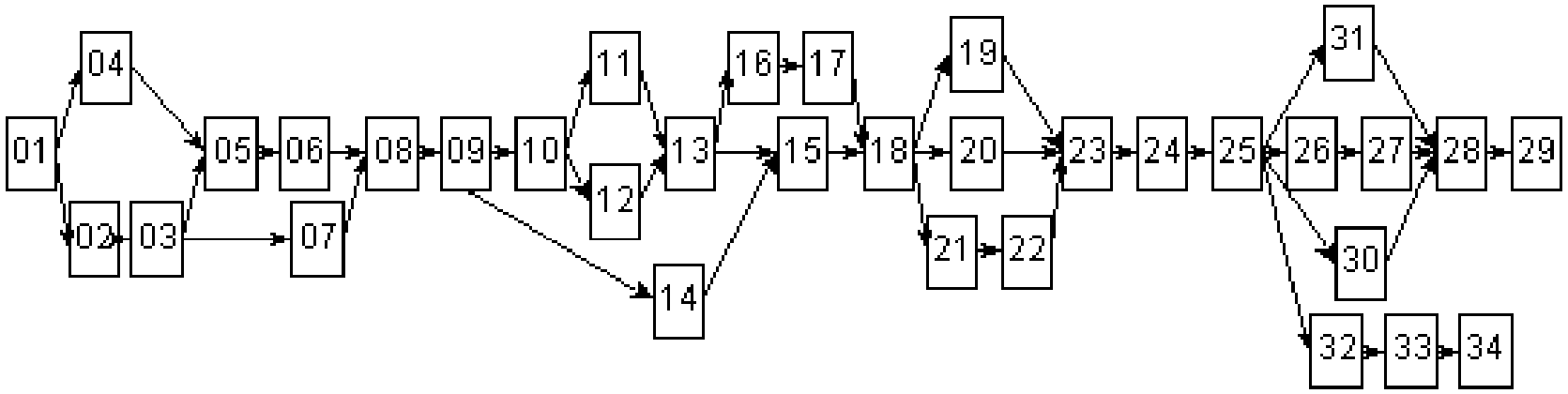
Task nr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Duration	3	4	4	6	5	2	3	4	3	2	3	2	1	5	2	3	2
Resource Use	4	4	3	4	5	5	4	3	4	8	4	5	4	3	3	3	6

Task nr	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Duration	2	1	1	1	2	4	5	2	1	1	2	1	3	2	1	2	2
Resource Use	7	4	4	4	4	7	8	8	3	3	6	8	3	3	3	3	3

- **Cumulative manpower constraint**
  - **different resource limits**

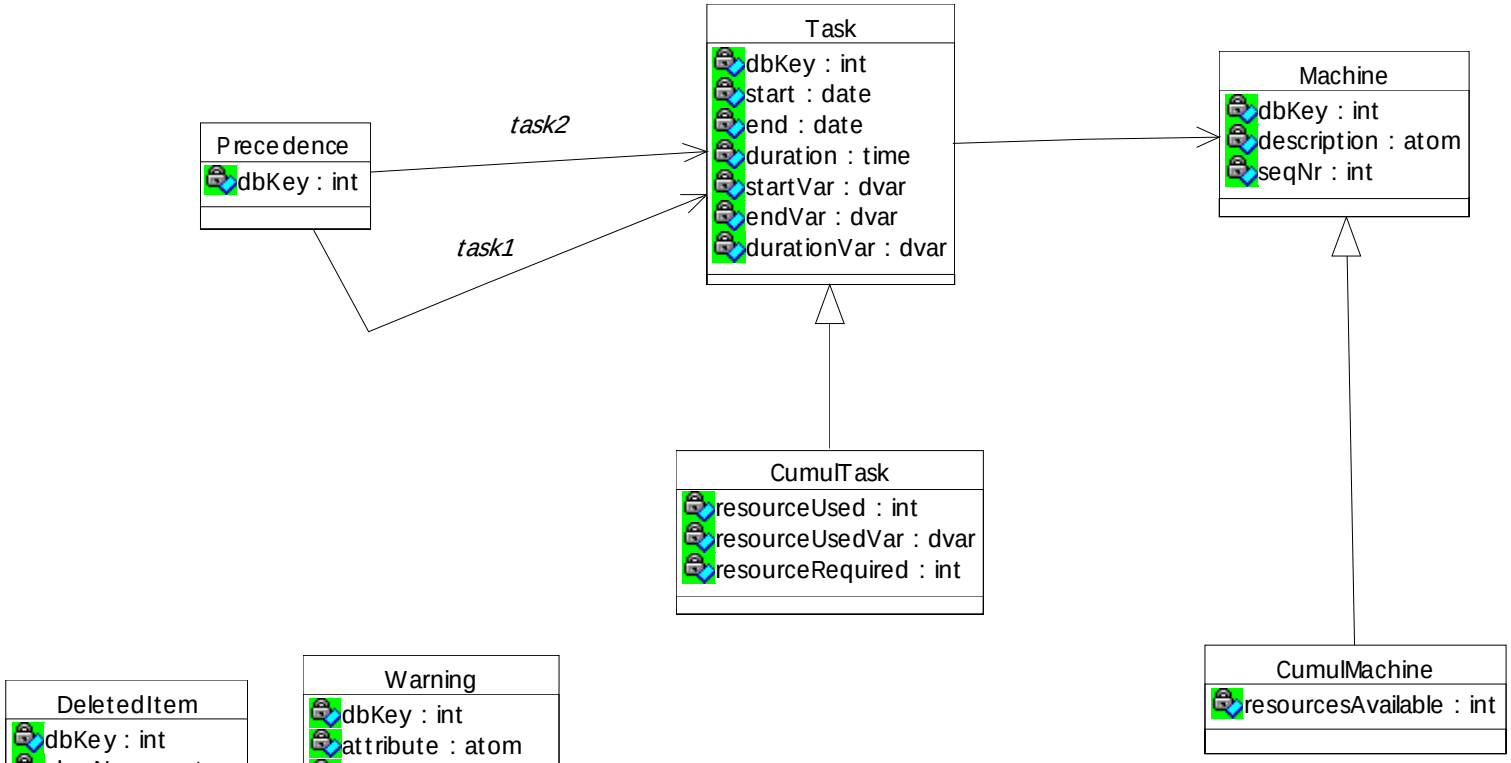
Problem Description (2)

- **Precedence constraints between tasks**

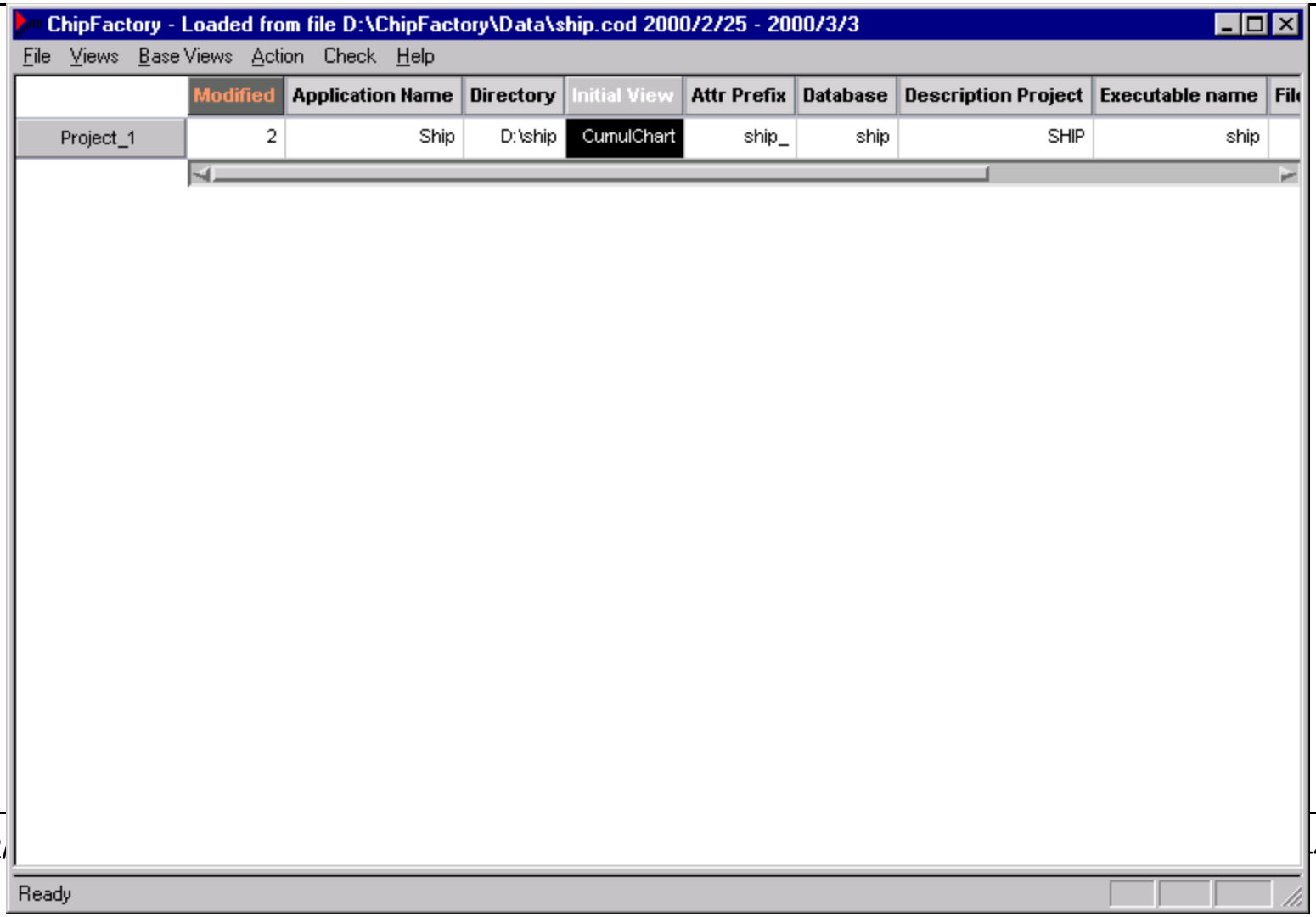


- **Minimize make-span**
  - **overall project end**
- **More detailed description in last year's tutorial**

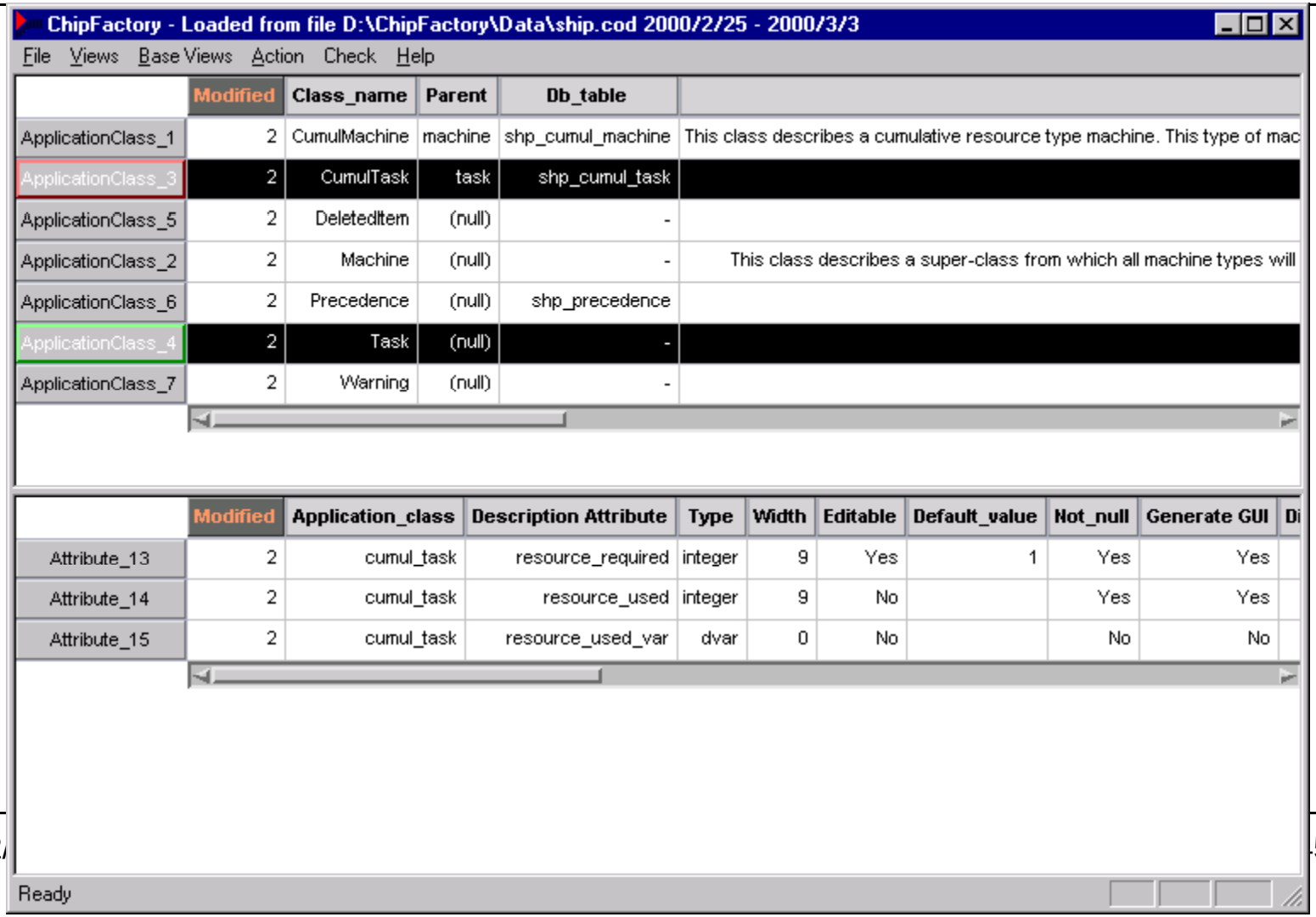
Object Model



Project Definition



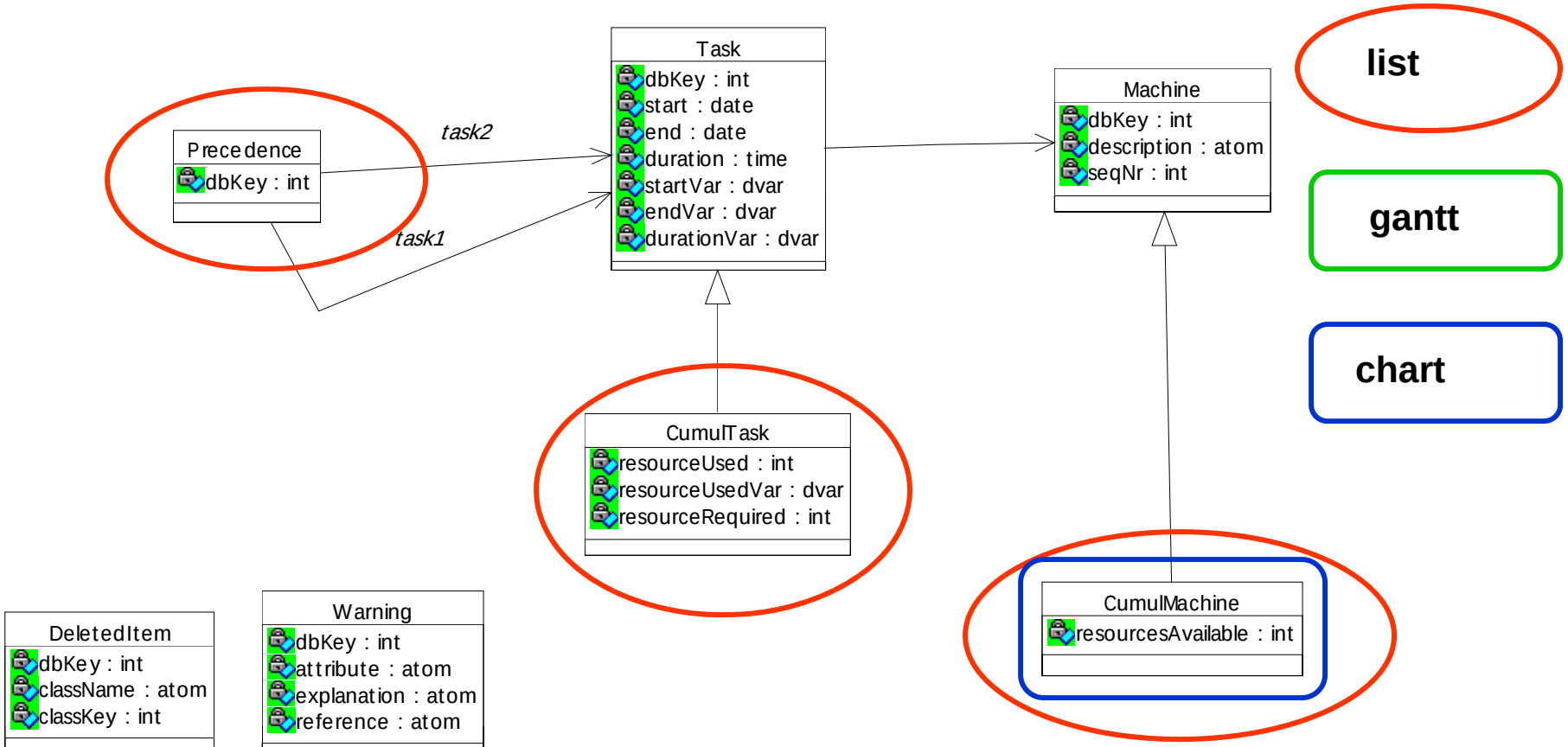
Class Definition



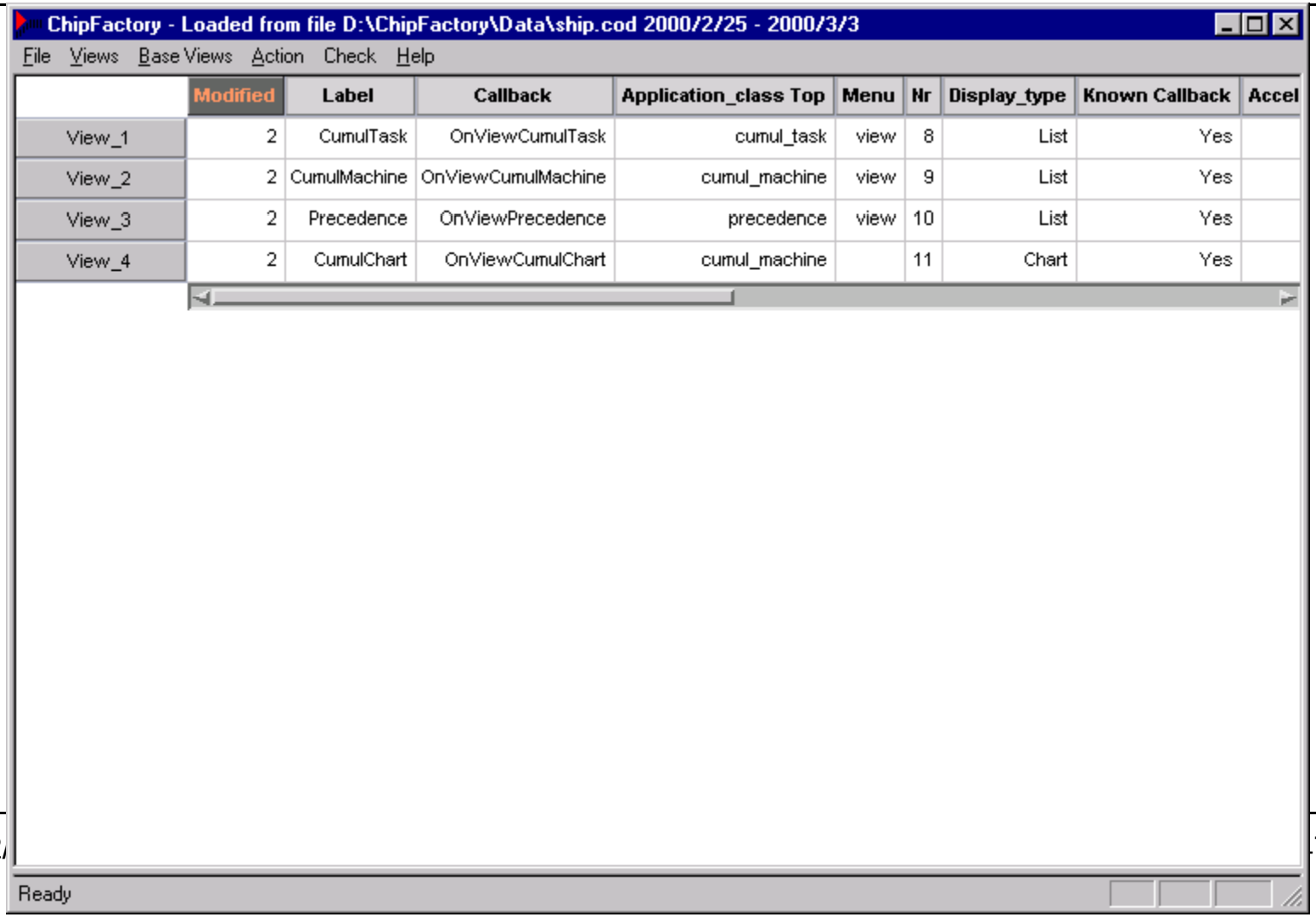
	Modified	Class_name	Parent	Db_table	
ApplicationClass_1	2	CumulMachine	machine	shp_cumul_machine	This class describes a cumulative resource type machine. This type of mac
ApplicationClass_3	2	CumulTask	task	shp_cumul_task	
ApplicationClass_5	2	DeletedItem	(null)	-	
ApplicationClass_2	2	Machine	(null)	-	This class describes a super-class from which all machine types will
ApplicationClass_6	2	Precedence	(null)	shp_precedence	
ApplicationClass_4	2	Task	(null)	-	
ApplicationClass_7	2	Warning	(null)	-	

	Modified	Application_class	Description Attribute	Type	Width	Editable	Default_value	Not_null	Generate GUI	Di
Attribute_13	2	cumul_task	resource_required	integer	9	Yes	1	Yes	Yes	
Attribute_14	2	cumul_task	resource_used	integer	9	No		Yes	Yes	
Attribute_15	2	cumul_task	resource_used_var	dvar	0	No		No	No	

GUI Design

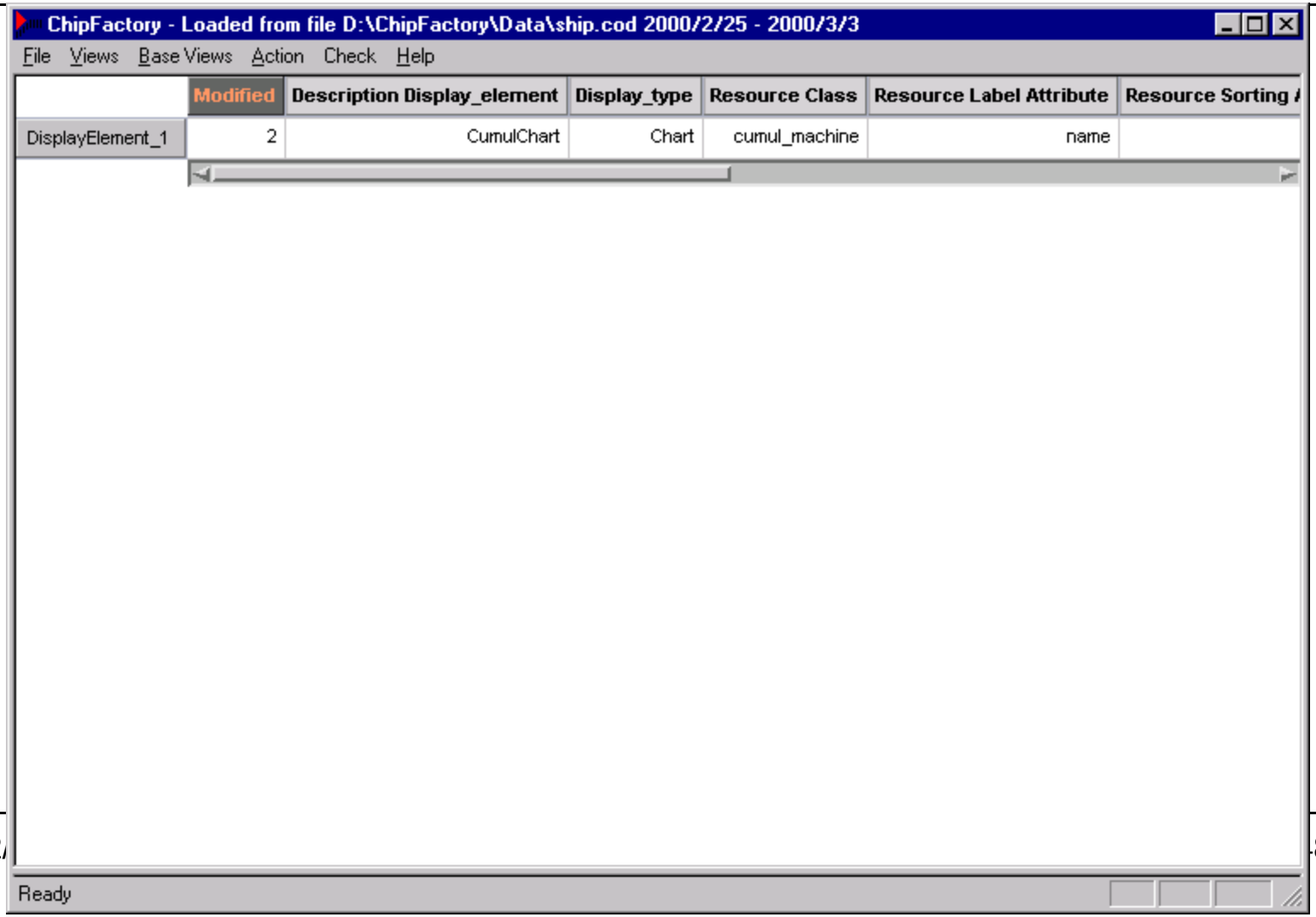

Task
dbKey : int
start : date
end : date
duration : time
startVar : dvar
endVar : dvar
durationVar : dvar
Machine
dbKey : int
description : atom
seqNr : int
Precedence
dbKey : int
task2
task1
CumulTask
resourceUsed : int
resourceUsedVar : dvar
resourceRequired : int
CumulMachine
resourcesAvailable : int
DeletedItem
dbKey : int
className : atom
classKey : int
Warning
dbKey : int
attribute : atom
explanation : atom
reference : atom
list
gantt
chart

GUI Definition

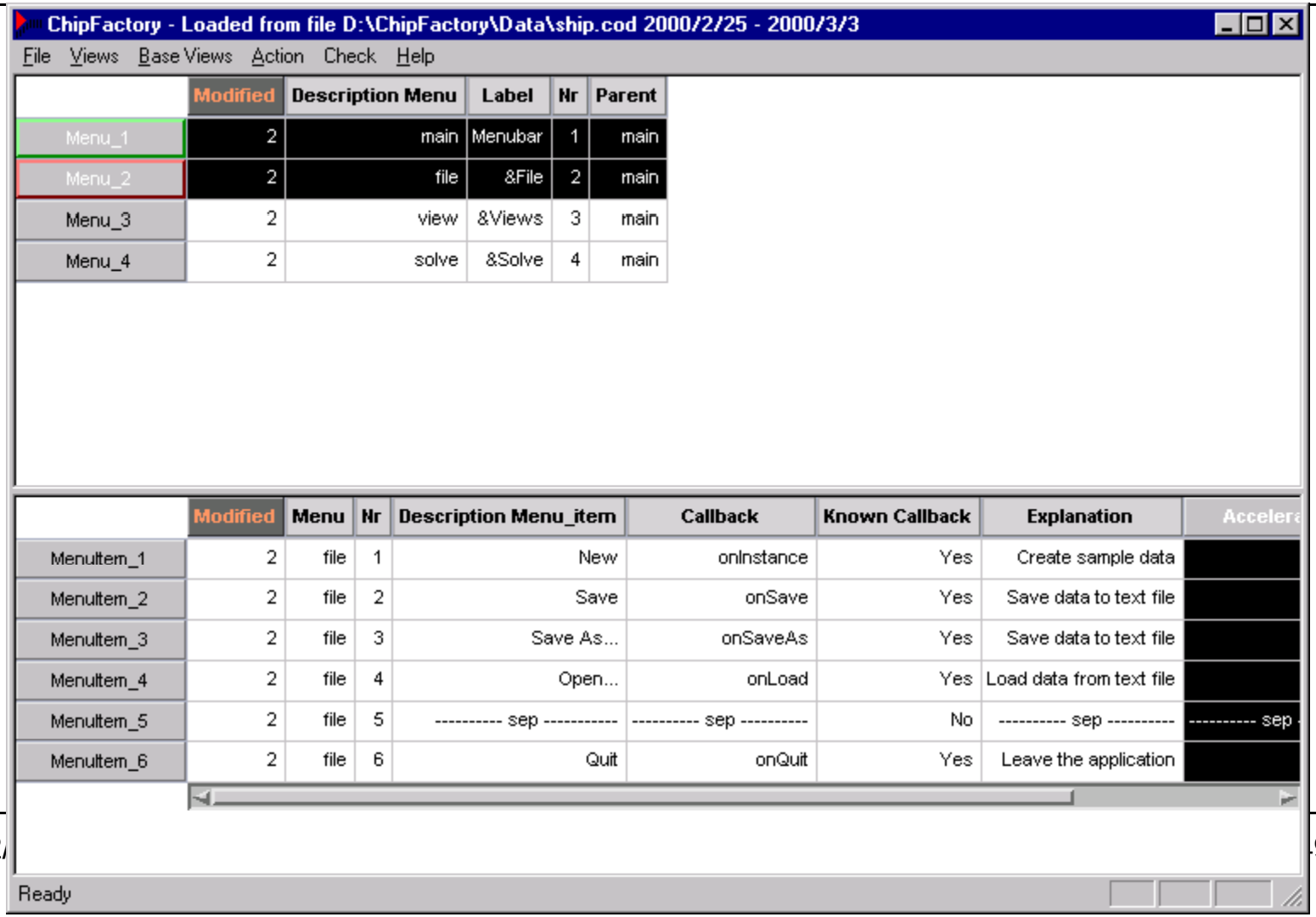


	Modified	Label	Callback	Application_class Top	Menu	Nr	Display_type	Known Callback	Accel
View_1	2	CumulTask	OnViewCumulTask	cumul_task	view	8	List	Yes	
View_2	2	CumulMachine	OnViewCumulMachine	cumul_machine	view	9	List	Yes	
View_3	2	Precedence	OnViewPrecedence	precedence	view	10	List	Yes	
View_4	2	CumulChart	OnViewCumulChart	cumul_machine		11	Chart	Yes	

Display Elements



Menu Definition



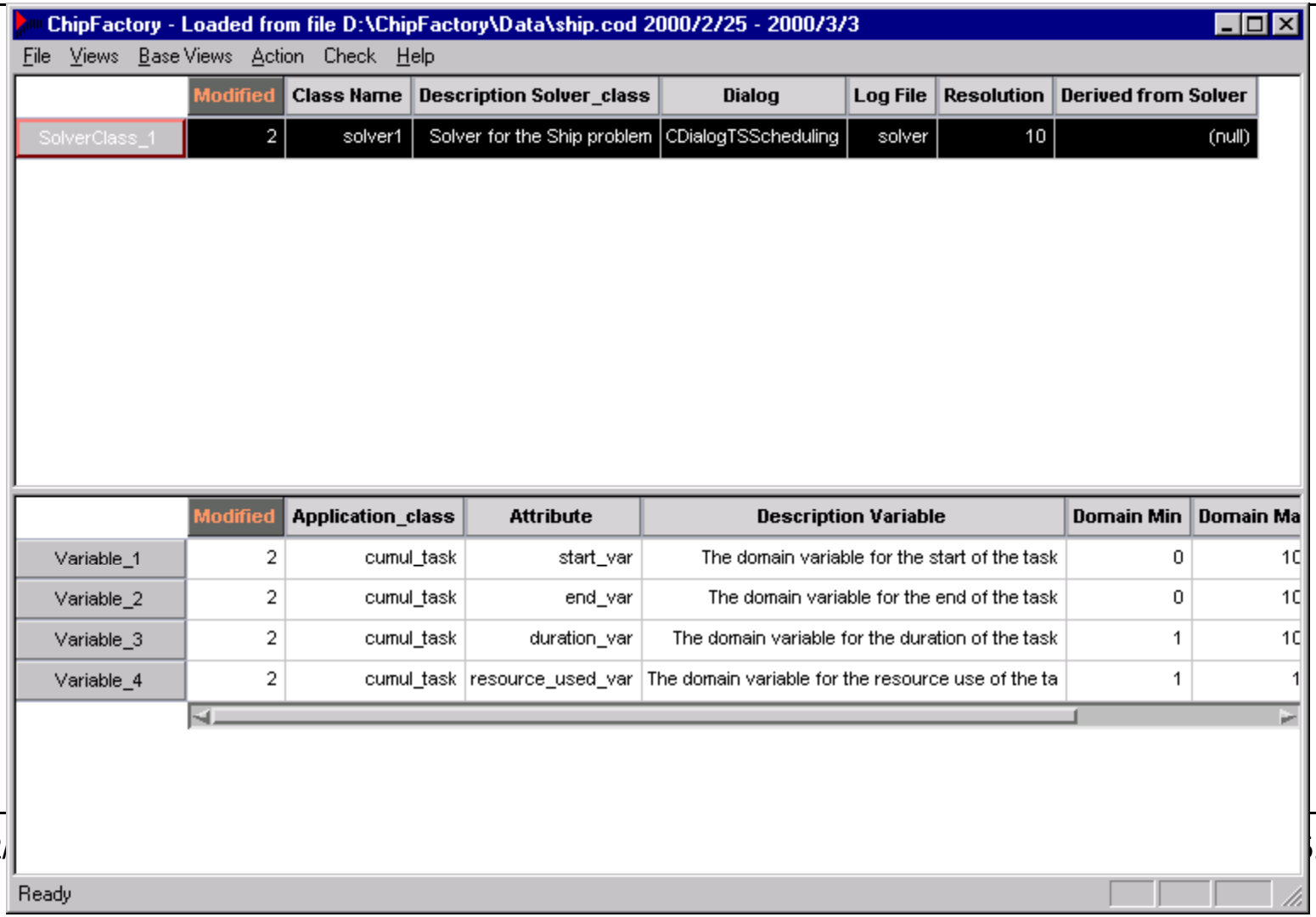
	Modified	Description Menu	Label	Nr	Parent
Menu_1	2	main	Menubar	1	main
Menu_2	2	file	&File	2	main
Menu_3	2	view	&Views	3	main
Menu_4	2	solve	&Solve	4	main

	Modified	Menu	Nr	Description Menu_item	Callback	Known Callback	Explanation	Accelera
MenuItem_1	2	file	1	New	onInstance	Yes	Create sample data	
MenuItem_2	2	file	2	Save	onSave	Yes	Save data to text file	
MenuItem_3	2	file	3	Save As...	onSaveAs	Yes	Save data to text file	
MenuItem_4	2	file	4	Open...	onLoad	Yes	Load data from text file	
MenuItem_5	2	file	5	---------- sep ----------	---------- sep ----------	No	---------- sep ----------	---------- sep
MenuItem_6	2	file	6	Quit	onQuit	Yes	Leave the application	

Solver1 Class

```
class Solver1 :public Solver
{
public:
        Solver1(LxTime start,LxTime end, Boolean silent,char *logFile);
        virtual void initialize();
        virtual void createVariables();
        virtual void createConstraints();
        virtual void enumerate();
        virtual void terminate();
        ChipResult cumulTaskEnumerate();
        ChipBool cumulTaskEnumerate(CumulTask **tasks, int i, int nbTasks, int remain);
        ChipBool cumulTaskMinMax(CumulTask **tasks, int nbTasks);
        ChipBool cumulTaskMinimize(CumulTask **tasks, int nbTasks);
        ChipBool cumulTaskMinimize(CumulTask **tasks, int i, int nbTasks, int remain);
        void cumulTaskSaveSolution();
        void cumulTaskRecallSolution();
        void cumulTaskResult();
        void cumulTaskSwapTask(CumulTask **tasks, int i, int j);
        int cumulTaskFindBest(CumulTask **tasks, int i, int nbTasks);
        Boolean cumulTaskBetterTask(CumulTask *x, CumulTask *y);
}
```

Variable Definition



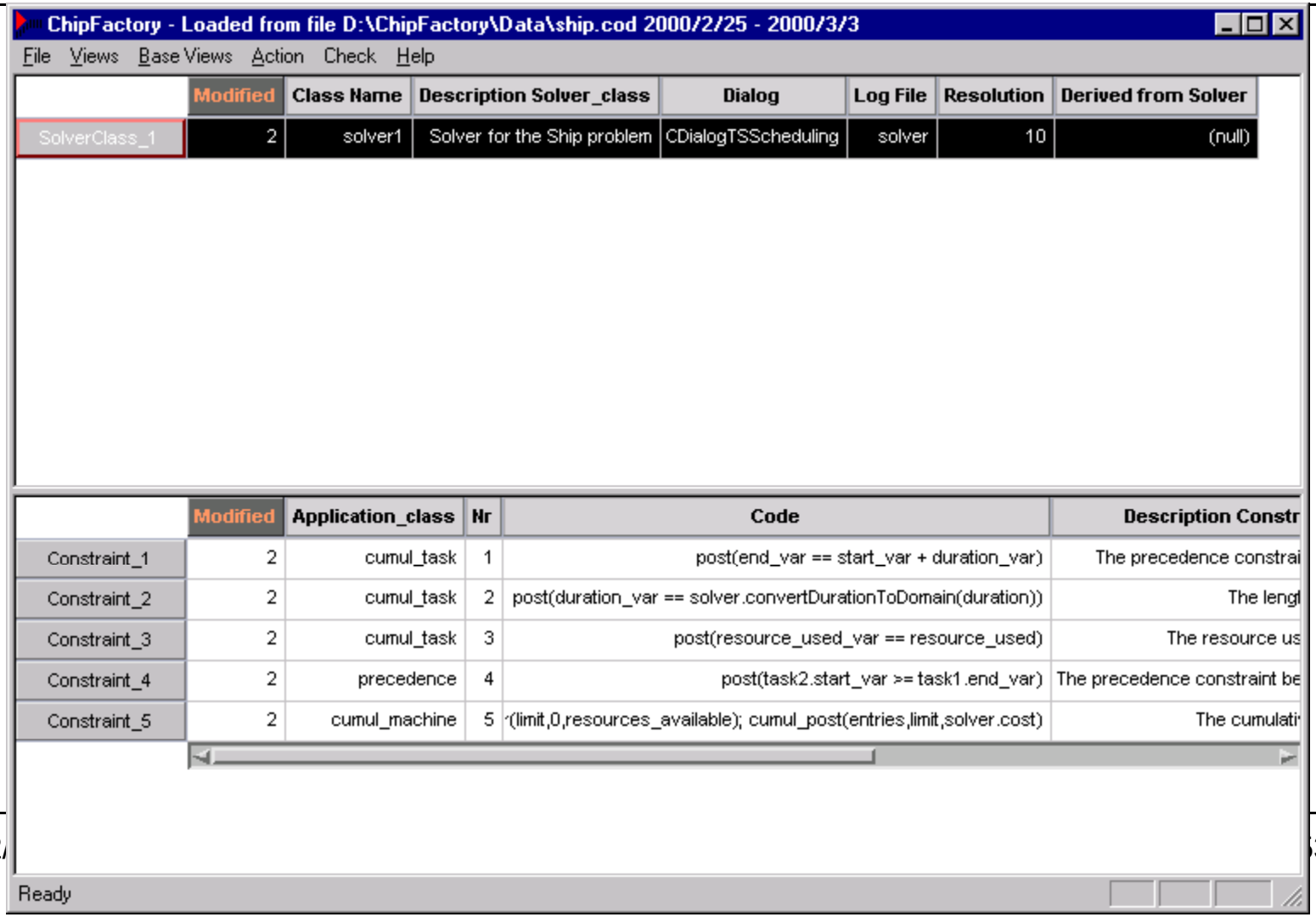
	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the Ship problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Attribute	Description Variable	Domain Min	Domain Ma
Variable_1	2	cumul_task	start_var	The domain variable for the start of the task	0	10
Variable_2	2	cumul_task	end_var	The domain variable for the end of the task	0	10
Variable_3	2	cumul_task	duration_var	The domain variable for the duration of the task	1	10
Variable_4	2	cumul_task	resource_used_var	The domain variable for the resource use of the ta	1	1

Generated Code

```
void Solver1::createVariables()
{
    ChipDvar cost(0,100000);
    setCost(cost);
    FORALL(x1, CumulTask, 1) {
        x1->initDomainVar(this);
    }
}
void CumulTask::initDomainVar(Solver1 *solver)
{
    ChipDvar v1 = ChipDvar(0, 100);setStartVar(v1);
    ChipDvar v2 = ChipDvar(0, 100);setEndVar(v2);
    ChipDvar v3 = ChipDvar(1, 100);setDurationVar(v3);
    ChipDvar v4 = ChipDvar(1, 10);setResourceUsedVar(v4);
}
```

Constraint Definition



	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the Ship problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Nr	Code	Description Constr
Constraint_1	2	cumul_task	1	post(end_var == start_var + duration_var)	The precedence constrai
Constraint_2	2	cumul_task	2	post(duration_var == solver.convertDurationToDomain(duration))	The lengt
Constraint_3	2	cumul_task	3	post(resource_used_var == resource_used)	The resource us
Constraint_4	2	precedence	4	post(task2.start_var >= task1.end_var)	The precedence constraint be
Constraint_5	2	cumul_machine	5	(limit,0,resources_available); cumul_post(entries,limit,solver.cost)	The cumulati

Constraint Model

- **CumulTask**

  post(end_var == start_var + duration_var)

  post(duration_var == solver.convertDurationToDomain(duration))

  post(resource_used_var == resource_used)

- **Precedence**

  post(task2.start_var >= task1.end_var)

- **CumulMachine**

```
cumul_entries(entries);
forall(x,cumul_task,
        if (x.resource == this) then
        entries.add(x.start_var,x.duration_var,x.resource_used_var)
) ;
dvar(limit,0,resources_available);
cumul_post(entries,limit,solver.cost)
```

Generated Code

```
void Solver1::createConstraints()
{
    solverStatusMessage("CumulTask constraints");
    FORALL(x1, CumulTask, 1) {
        x1->defineConstraint(this);
    }
    solverStatusMessage("Precedence constraints");
    FORALL(x2, Precedence, 2) {
        x2->defineConstraint(this);
    }
    solverStatusMessage("CumulMachine constraints");
    FORALL(x3, CumulMachine, 3) {
        x3->defineConstraint(this);
    }
}
```

Generated Code: CumulTask

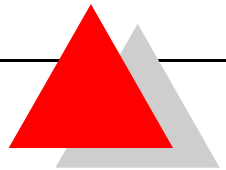
```
void CumulTask::defineConstraint(Solver1 *solver)
{
    if (ChipPost(getEndVar() == getStartVar() + getDurationVar())
                        != ChipSucceed) {
            solver->solverError("Constraint failed");
    };
    if (ChipPost(getDurationVar() ==
                        solver->convertDurationToDomain(getDuration()))
                         != ChipSucceed) {
            solver->solverError("Constraint failed");
    };
    if (ChipPost(getResourceUsedVar() == getResourceUsed())
                        != ChipSucceed) {
            solver->solverError("Constraint failed");
    };
}
```

Generated Code: CumulMachine

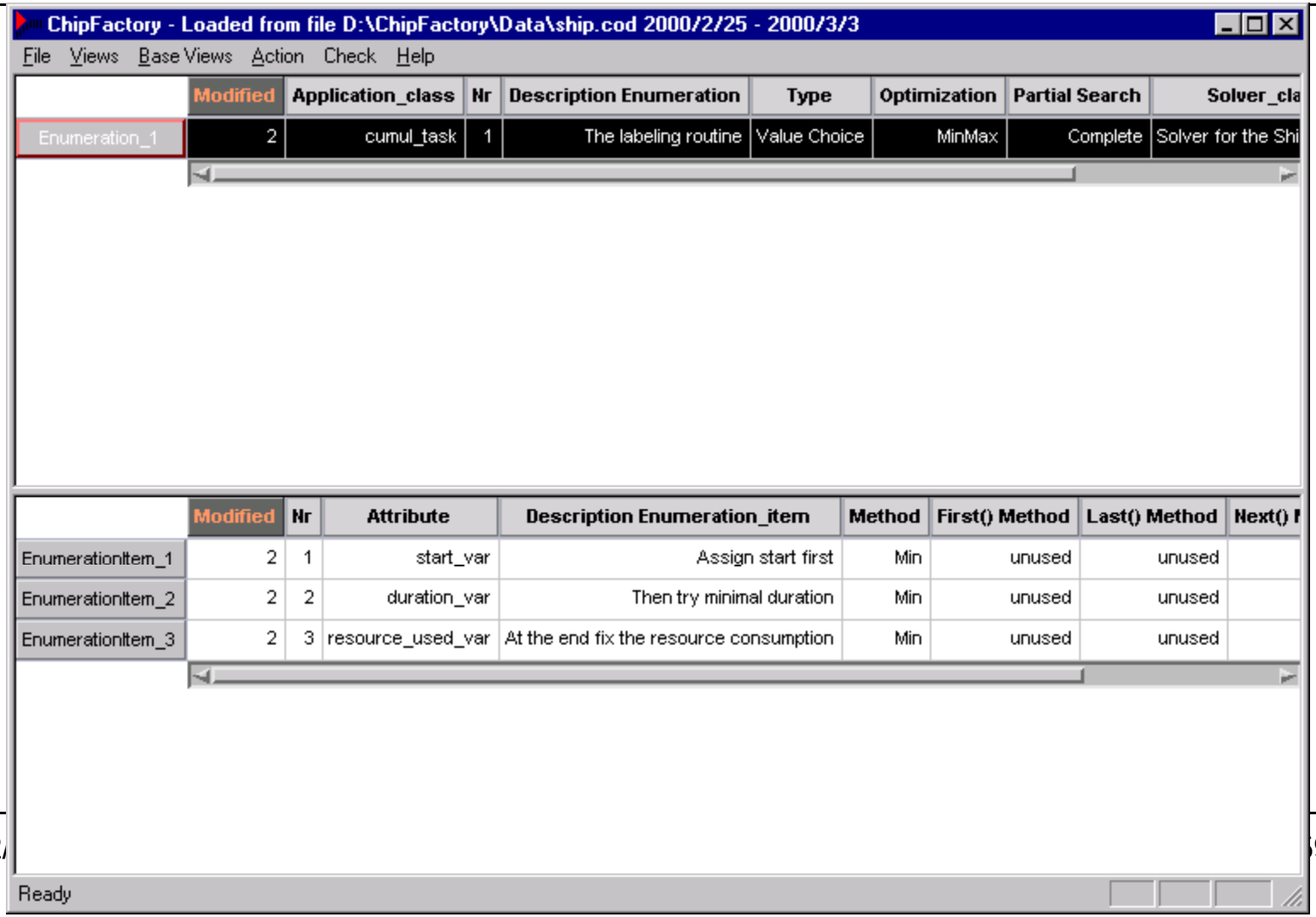
```
void CumulMachine::defineConstraint(Solver1 *solver)
{
    ChipCumEntries entries;
    FORALL(x,CumulTask,1) {
        if (x->getResource() == this) {
            entries.append(ChipCumEntry(x->getStartVar(),
            x->getDurationVar(), x->getResourceUsedVar()));
        };
    };
    ChipDvar limit(0, getResourcesAvailable());
    if (!entries.isEmpty()) {
        ChipCumulative cumul(entries,limit);
        cumul.setEnd(solver->getCost());
        if (ChipPost(cumul) != ChipSucceed) {
            solver->solverError("Constraint failed");
        };
    };
}
```

Generated Code: Precedence

```
void Precedence::defineConstraint(Solver1 *solver)
{
    if (ChipPost(getTask2()->getStartVar() >= getTask1()->getEndVar())
                        != ChipSucceed) {
            solver->solverError("Constraint failed");
    };
}
```



Enumeration Definition



Basic Enumeration Routine (Generated code)

```
ChipBool Solver1::cumulTaskEnumerate(CumulTask **tasks, int i, int nbTasks, int remain)
{
	percentageStatus(nbTasks-remain,nbTasks);
	if (remain <= 0) {return ChipTrue;}
	CumulTask *task = (CumulTask *)tasks[i];
	ChipChoice startVarChoice;ChipDvar startVar = task->getStartVar();
		for(int val1 = startVar.min(); val1 >= 0; val1 = startVar.next(val1)) {
		startVarChoice.ChipRemember();
		if (ChipPost(startVar == val1) == ChipSucceed) {
				if (cumulTaskEnumerate(tasks,i+1,nbTasks, remain-1)) {
					return ChipTrue;
				}
		}
		incNbBacktrack();
		startVarChoice.ChipUndo();
	}
	return ChipFalse;
}
```

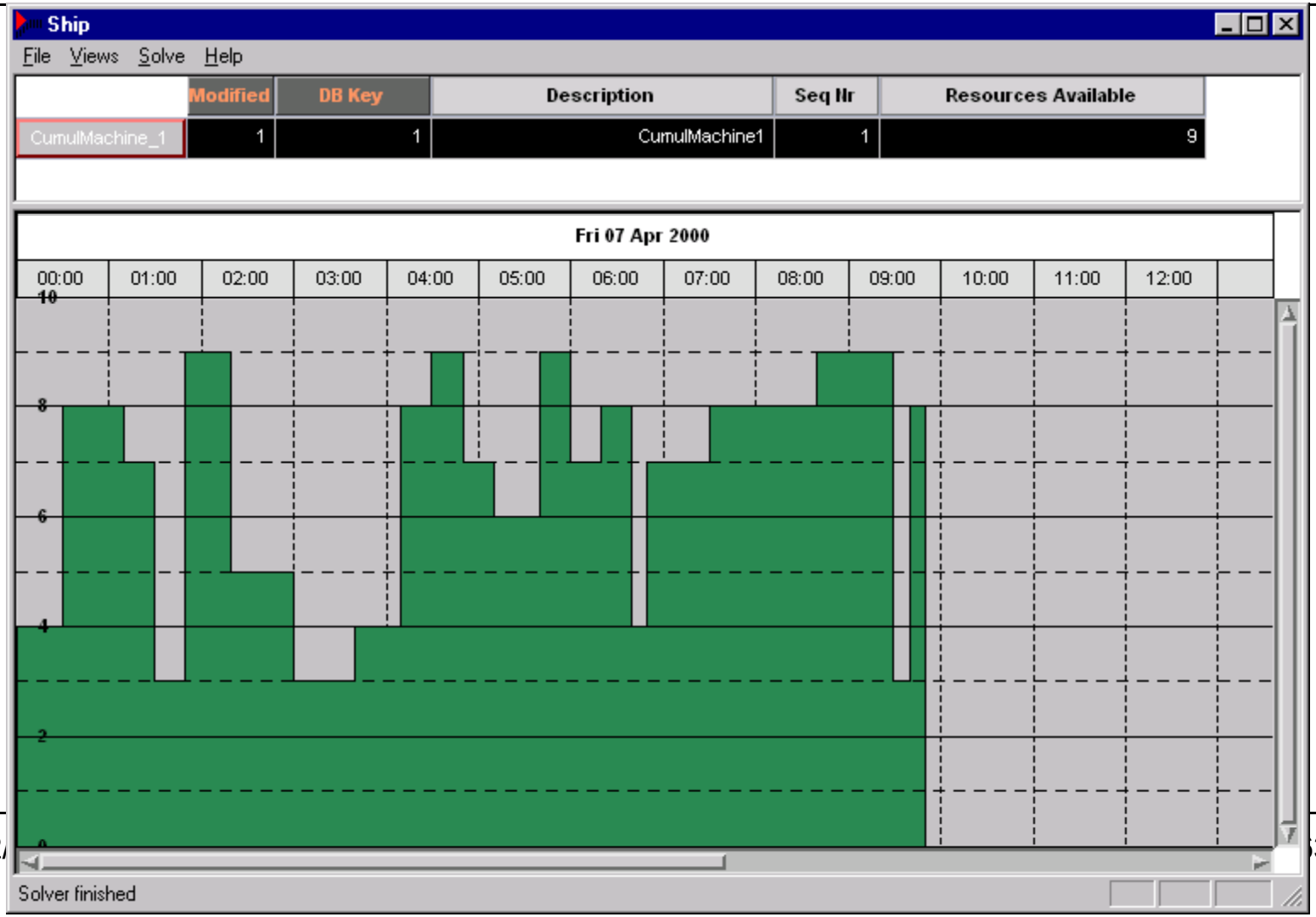
MinMax Optimization (Generated Code)

```
ChipBool Solver1::cumulTaskMinMax(CumulTask **tasks, int n)
{
	ChipChoice begin;
	begin.ChipRemember();
	while (!done) {
		begin.ChipUndo();
		if (ChipPost(getCost() < saved_cost) == ChipSucceed &&
	cumulTaskEnumerate(tasks,1,n,n)) {
			cumulTaskSaveSolution();
		} else done = TRUE;
	}
	begin.ChipUndo();
	if (found) {
		cumulTaskRecallSolution();
		return ChipTrue;
	} else return ChipFalse;
}
```

Minimize Optimization (Generated code)

```
ChipBool Solver1::cumulTaskMinimize(CumulTask **tasks, int n)
{
	ChipChoice begin;
	begin.ChipRemember();
	try{
		cumulTaskMinimize(tasks,1,n,n);
	}
	catch(const char *msg) {}
	begin.ChipUndo();
	if (found) {
		cumulTaskRecallSolution();
		return ChipTrue;
	} else {
		return ChipFalse;
	}
}
```

Finished Application



Example 2

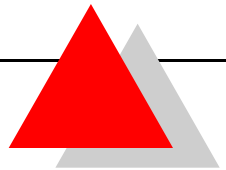
Chemical Process Scheduling

Problem Description

- **Scheduling problem with machine choice/machine speed**

Nr	Mach 1	Mach 2	Mach 3	Mach 4
1	1538	-	-	1194
2	1500	-	-	789
3	1607	-	-	818
4	-	-	1564	2143
5	-	-	736	1017
6	5263	-	-	3200
7	4865	-	3035	3214
8	-	-	1500	1440
9	-	-	1869	2459
10	-	1282	-	-
11	-	3750	-	3000
12	-	6796	7000	5600

- **Overall cumulative constraint limiting nr of parallel tasks**
- **Hard due-date for each task**
- **Cost is sum of earlyness**



Problem Description (2)

- **Original description**

  [PG95] J.M. Pinto, I.E. Grossmann

  Continuous Time Mixed Integer Linear Programming Model for Short Term Scheduling of Multistage Batch Plants, Ind. Eng. Chem. Res. 1995, 34, 3037-3051

  [PG97] J. M. Pinto, I.E. Grossmann

  A logic Based Approach to Scheduling Problems with Resource Constraints

  Computers Chem Engineering, Vol 21, No 8, pp 801-818, 1997

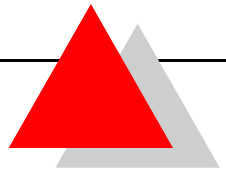
- **First CP model from V. Liatsos (IC-Parc)**
  - **described in**

    [ELS99] H. El Sakkout

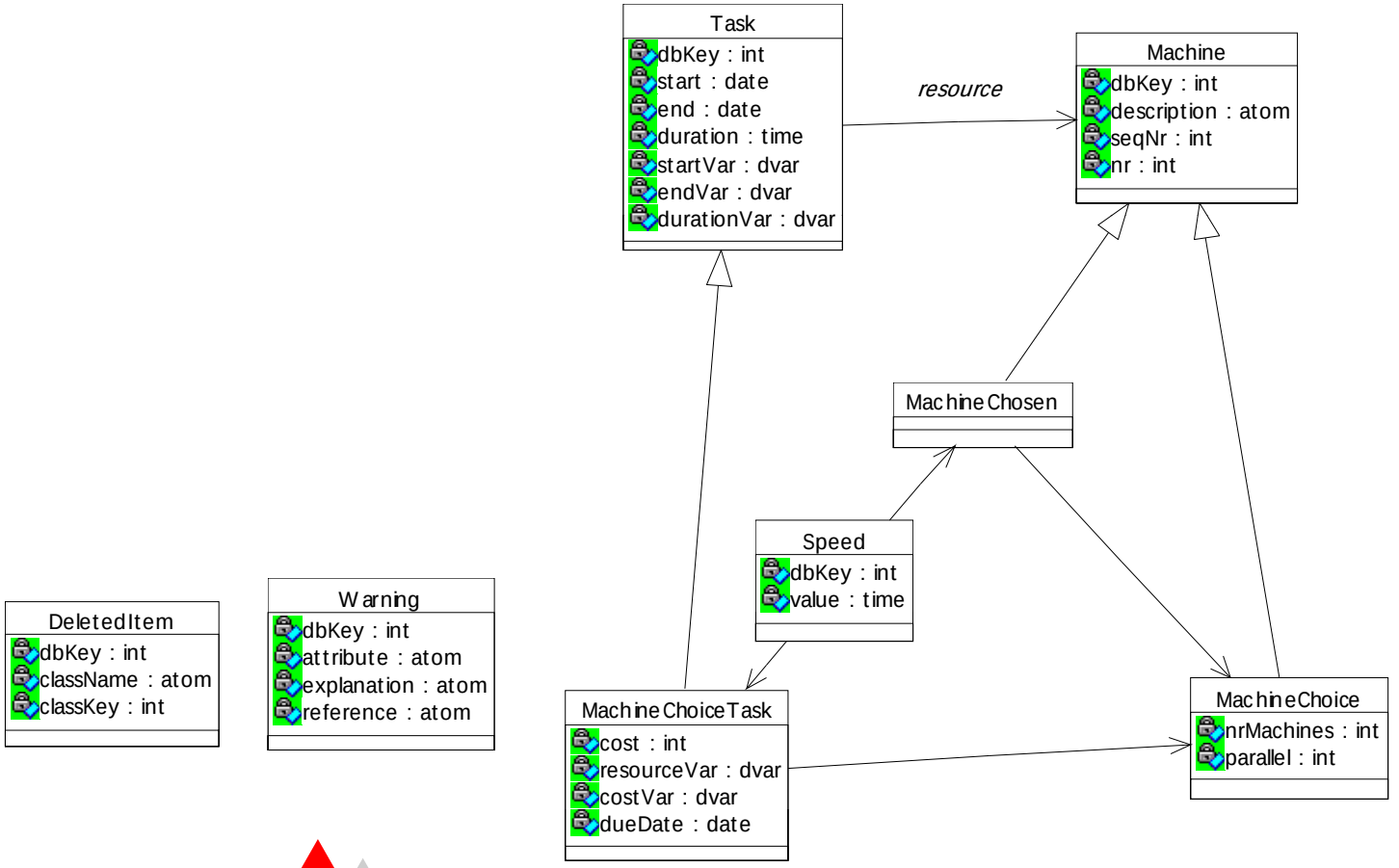
    Algorithm Hybridisation in Constraint Logic Programming

    Tutorial, PACLP 99, London, April 1999

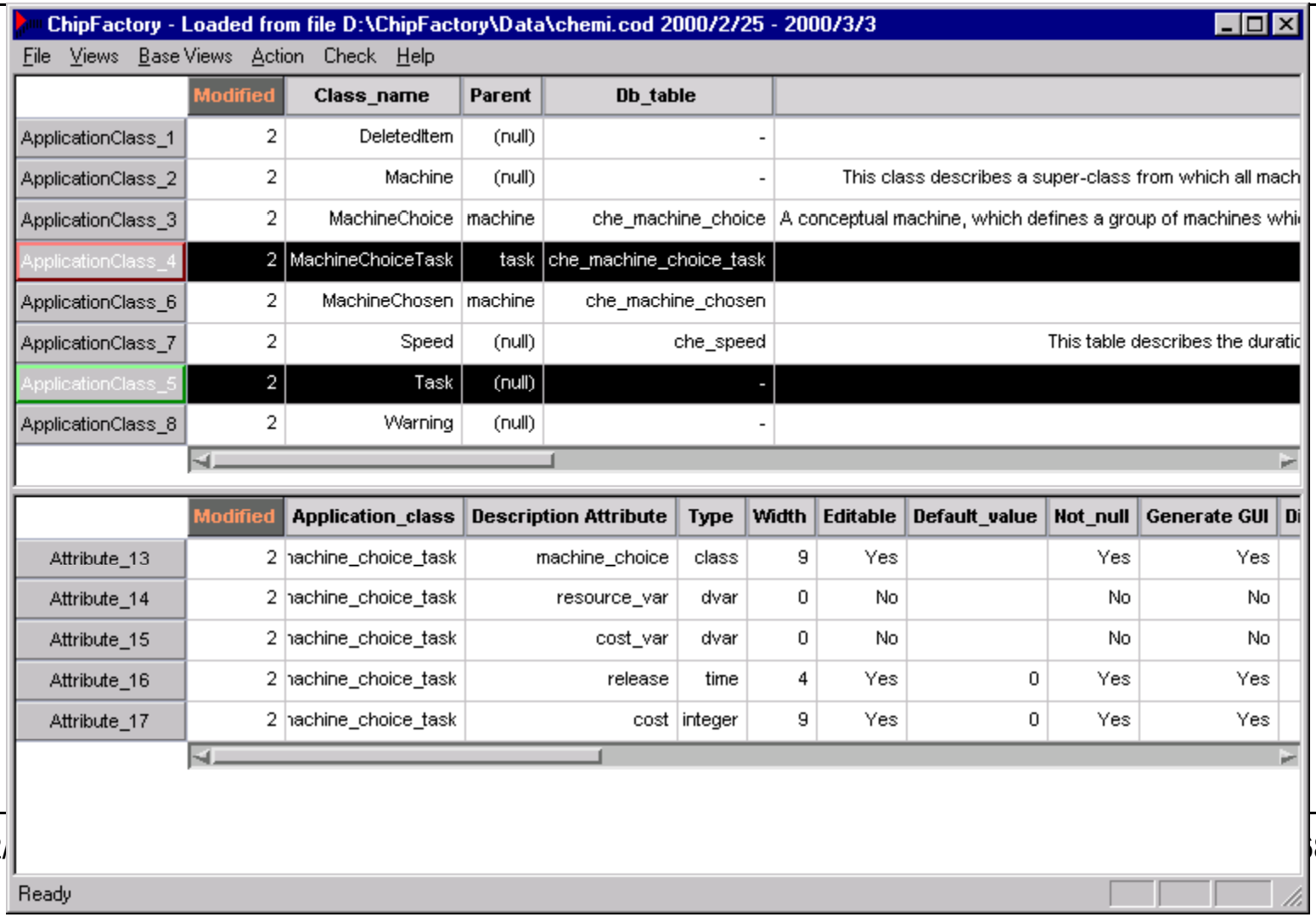
- **This model used here**



Object Model



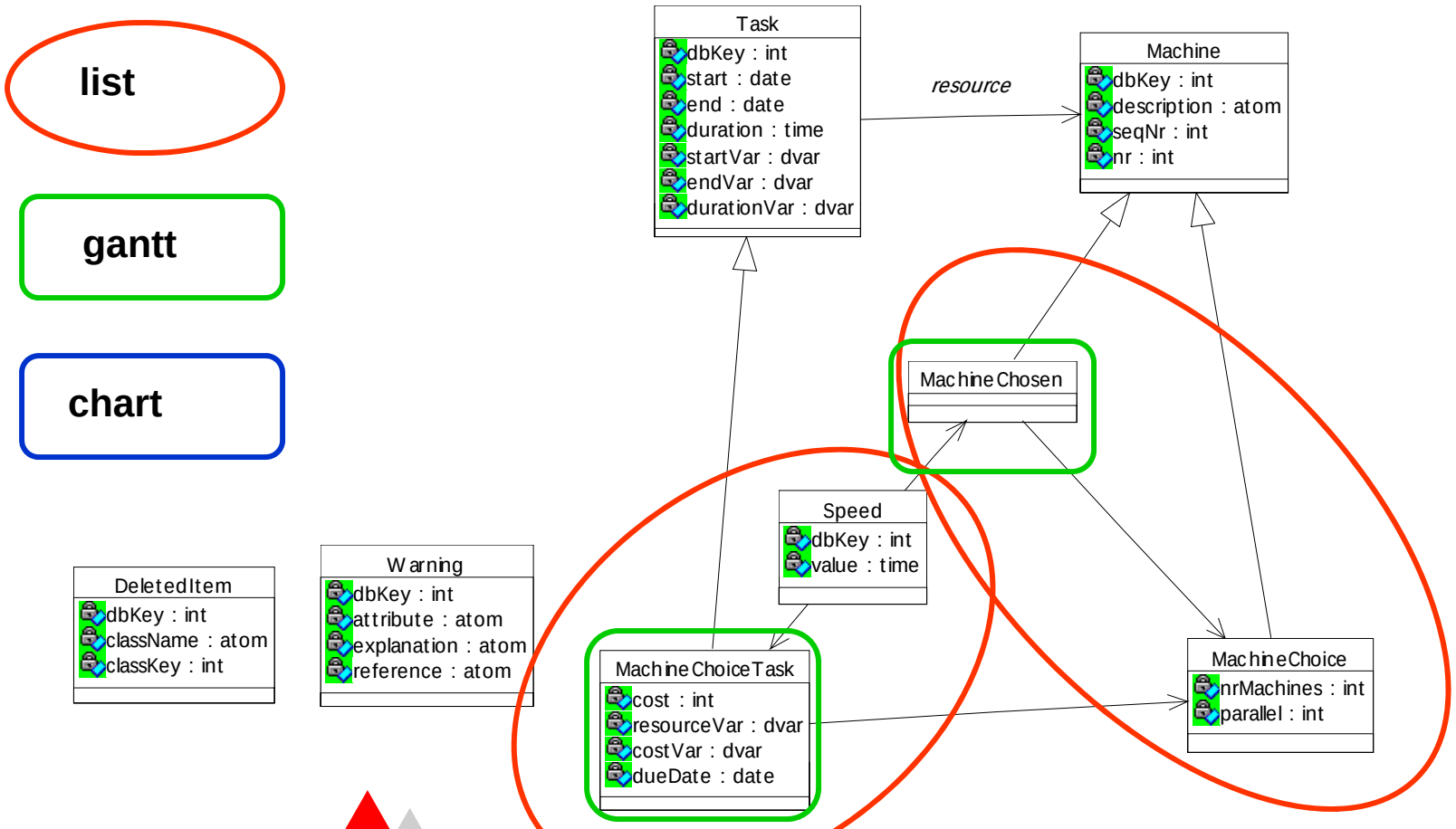
Object Definition



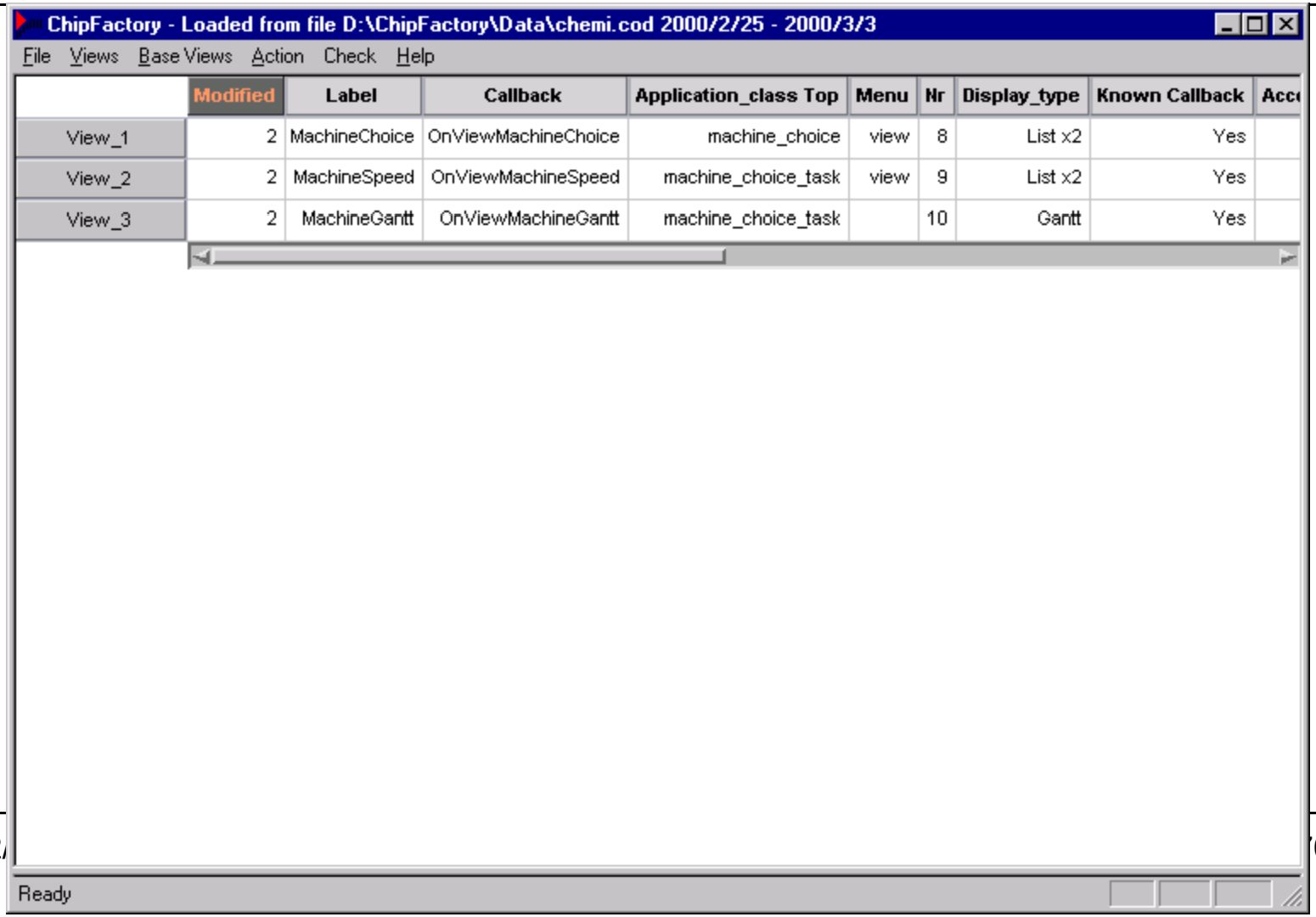
	Modified	Class_name	Parent	Db_table	
ApplicationClass_1	2	DeletedItem	(null)	-	
ApplicationClass_2	2	Machine	(null)	-	This class describes a super-class from which all mach
ApplicationClass_3	2	MachineChoice	machine	che_machine_choice	A conceptual machine, which defines a group of machines whi
ApplicationClass_4	2	MachineChoiceTask	task	che_machine_choice_task	
ApplicationClass_6	2	MachineChosen	machine	che_machine_chosen	
ApplicationClass_7	2	Speed	(null)	che_speed	This table describes the duratic
ApplicationClass_5	2	Task	(null)	-	
ApplicationClass_8	2	Warning	(null)	-	

	Modified	Application_class	Description Attribute	Type	Width	Editable	Default_value	Not_null	Generate GUI	Di
Attribute_13	2	achine_choice_task	machine_choice	class	9	Yes		Yes	Yes	
Attribute_14	2	achine_choice_task	resource_var	dvar	0	No		No	No	
Attribute_15	2	achine_choice_task	cost_var	dvar	0	No		No	No	
Attribute_16	2	achine_choice_task	release	time	4	Yes	0	Yes	Yes	
Attribute_17	2	achine_choice_task	cost	integer	9	Yes	0	Yes	Yes	

GUI Design

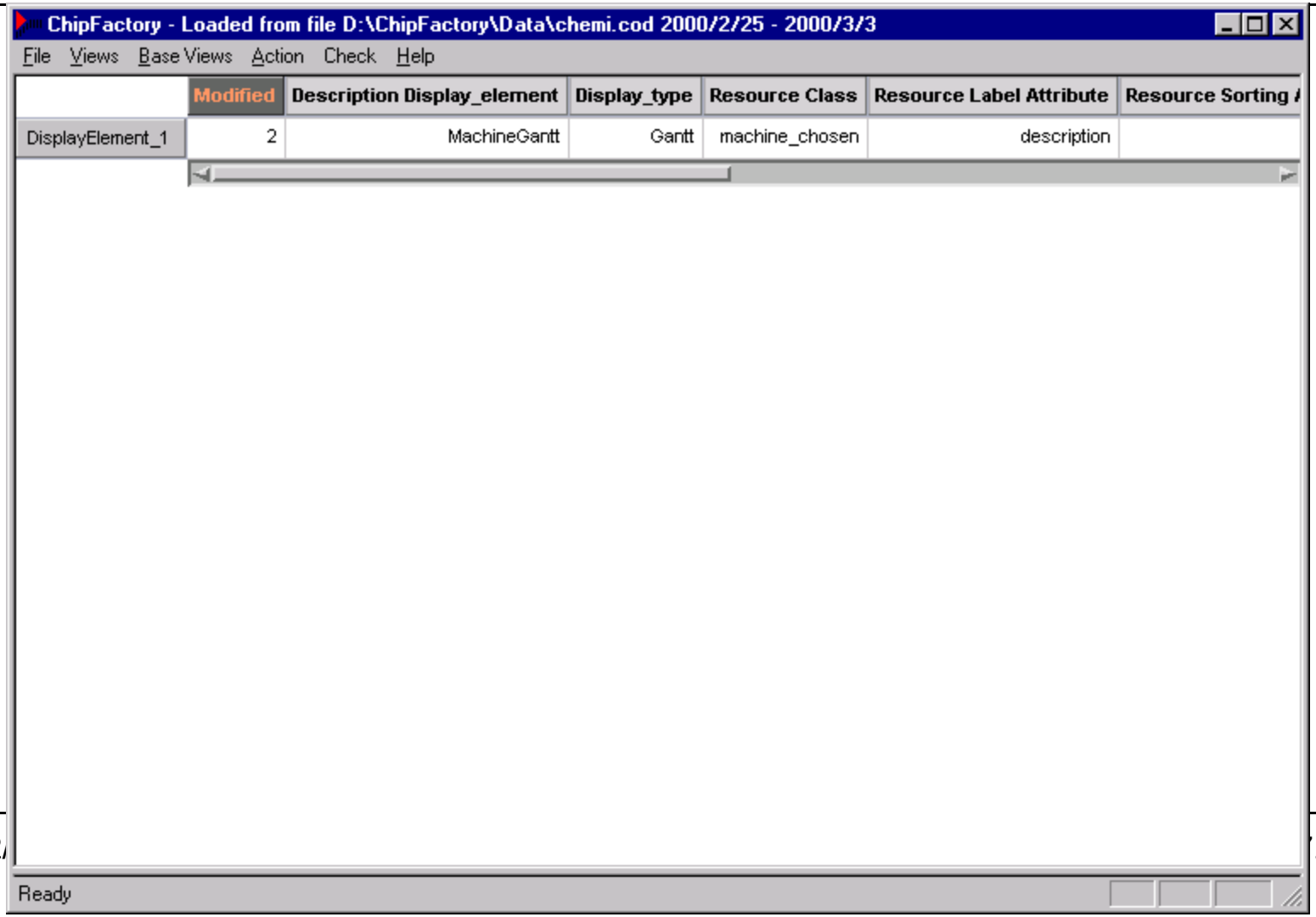

list
gantt
chart
Task
dbKey : int
start : date
end : date
duration : time
startVar : dvar
endVar : dvar
durationVar : dvar
resource
Machine
dbKey : int
description : atom
seqNr : int
nr : int
Machine Chosen
Speed
dbKey : int
value : time
DeletedItem
dbKey : int
className : atom
classKey : int
Warning
dbKey : int
attribute : atom
explanation : atom
reference : atom
Machine Choice Task
cost : int
resourceVar : dvar
costVar : dvar
dueDate : date
Machine Choice
nrMachines : int
parallel : int

GUI Definition

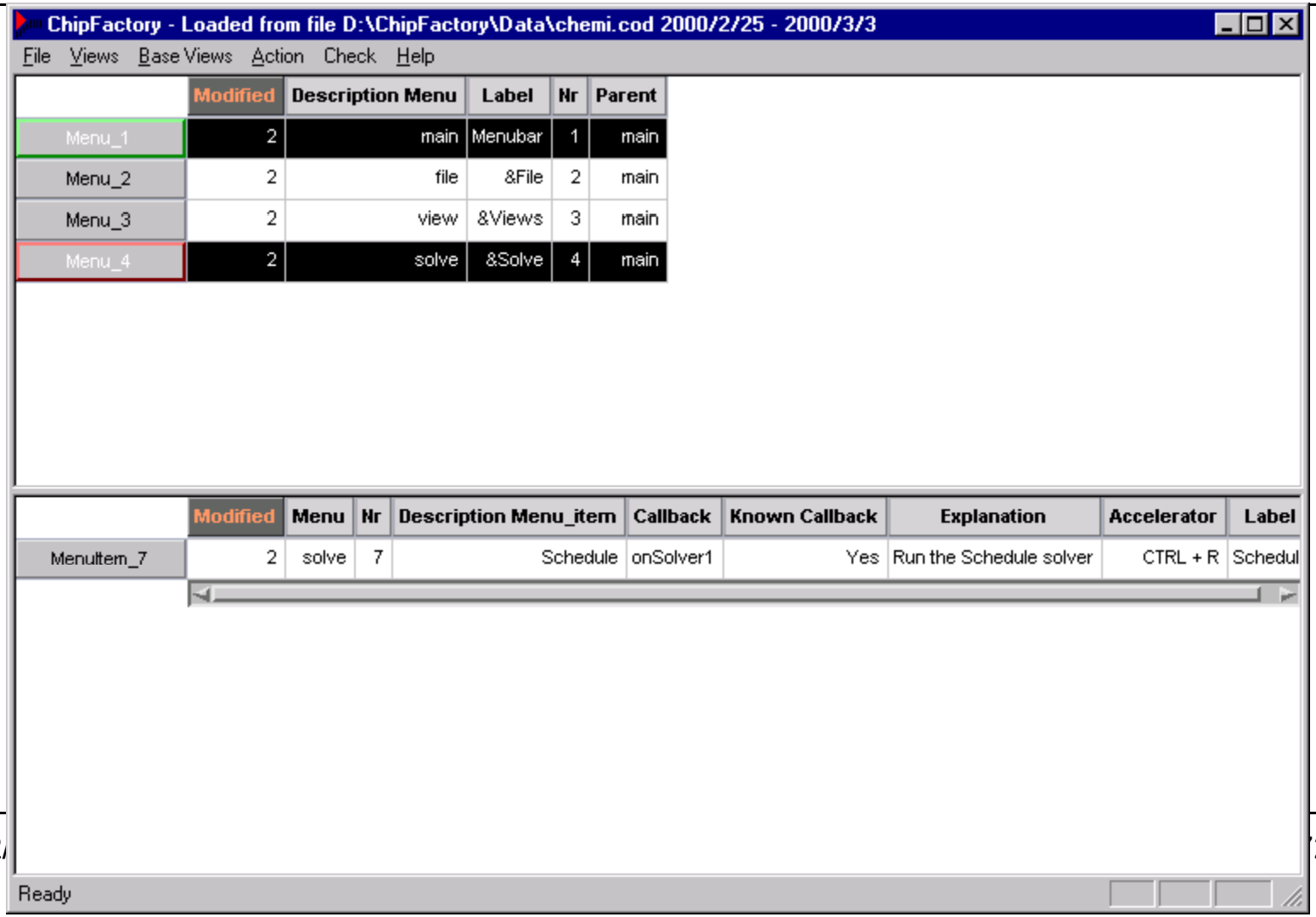


	Modified	Label	Callback	Application_class Top	Menu	Nr	Display_type	Known Callback	Acc
View_1	2	MachineChoice	OnViewMachineChoice	machine_choice	view	8	List x2	Yes	
View_2	2	MachineSpeed	OnViewMachineSpeed	machine_choice_task	view	9	List x2	Yes	
View_3	2	MachineGantt	OnViewMachineGantt	machine_choice_task		10	Gantt	Yes	

Display Elements



Menu Definition



Variable Definition

ChipFactory - Loaded from file D:\ChipFactory\Data\chemi.cod 2000/2/25 - 2000/3/3

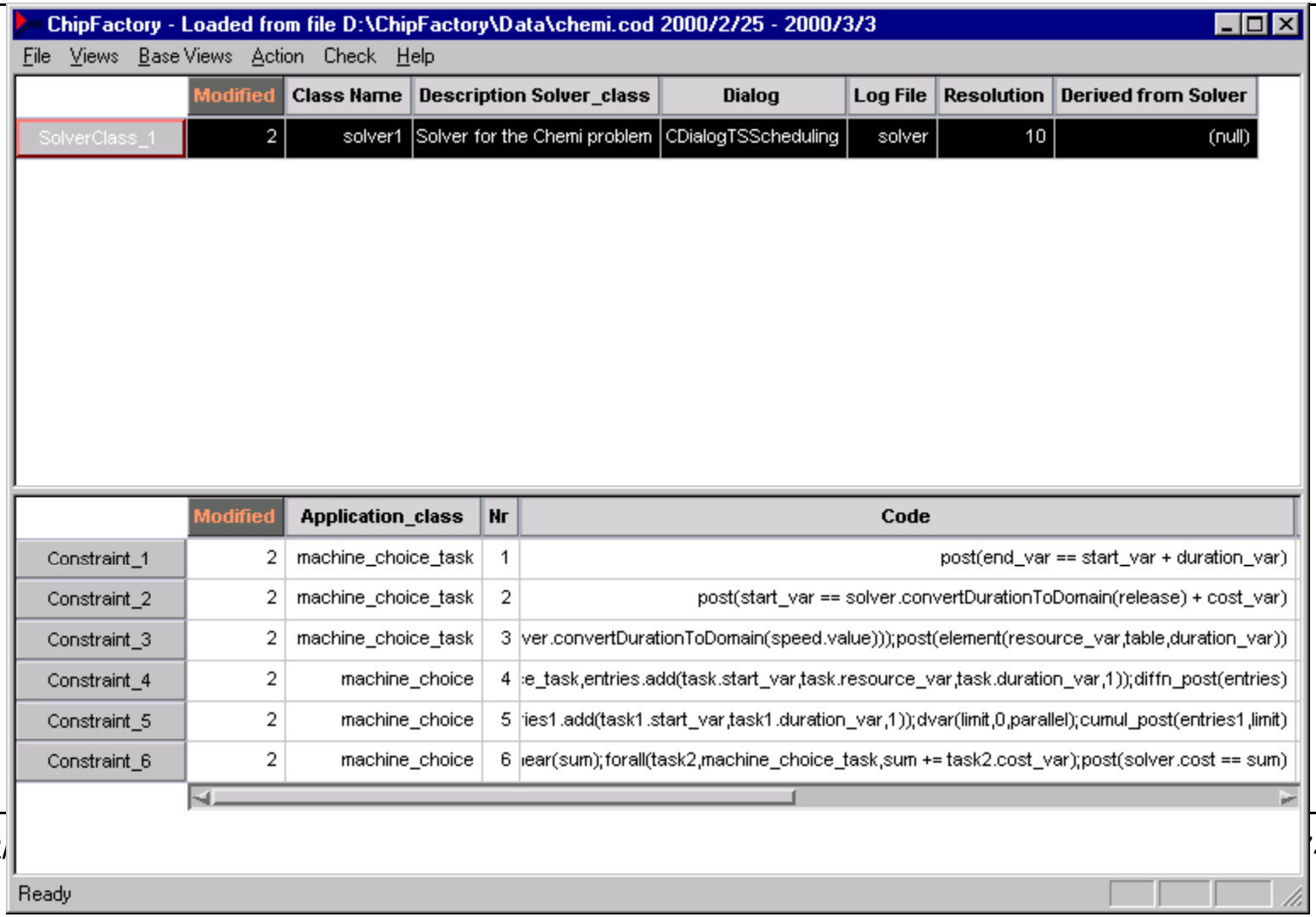
File Views Base Views Action Check Help

	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the Chemi problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Attribute	Description Variable	Domain Min	Domain Max	
Variable_1	2	machine_choice_task	start_var	The domain variable for the start of the task	0	20000	S
Variable_2	2	machine_choice_task	end_var	The domain variable for the end of the task	0	20000	S
Variable_3	2	machine_choice_task	duration_var	The domain variable for the duration of the task	1	20000	S
Variable_4	2	machine_choice_task	resource_var	The domain variable for the resource use of the ta	1	10	S
Variable_4	2	machine_choice_task	cost_var	The domain variable for the resource use of the ta	0	20000	S

Ready

Constraint Definition



	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the Chemi problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Nr	Code
Constraint_1	2	machine_choice_task	1	post(end_var == start_var + duration_var)
Constraint_2	2	machine_choice_task	2	post(start_var == solver.convertDurationToDomain(release) + cost_var)
Constraint_3	2	machine_choice_task	3	ver.convertDurationToDomain(speed.value)));post(element(resource_var,table,duration_var))
Constraint_4	2	machine_choice	4	e_task,entries.add(task.start_var,task.resource_var,task.duration_var,1));diffn_post(entries)
Constraint_5	2	machine_choice	5	ies1.add(task1.start_var,task1.duration_var,1));dvar(limit,0,parallel);cumul_post(entries1,limit)
Constraint_6	2	machine_choice	6	ear(sum);forall(task2,machine_choice_task,sum += task2.cost_var);post(solver.cost == sum)

Constraint Model

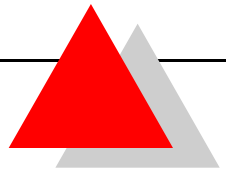
- **Task constraints between variables of task**
- **Machine Choice constraint**
- **Cost constraint as sum of individual costs**

Machine Choice Task Constraints

```
post(end_var == start_var + duration_var)
post(start_var + cost_var == solver.convertDurationToDomain(duedate))
int(table,machine_choice.nr_machines);
forall(speed,speed,
      if(speed.machine_choice_task == this,
            table(speed.machine_chosen.nr) = solver.convertDurationToDomain(speed.value)
      )
);
post(element(resource_var,table,duration_var))
```

Machine Choice Constraint

```
diffn_entries(entries);
forall(task,machine_choice_task,
      entries.add(task.start_var,task.resource_var,task.duration_var,1)
);
diffn_post(entries)
cumul_entries(entries1);
forall(task1,machine_choice_task,
      entries1.add(task1.start_var,task1.duration_var,1)
);
dvar(limit,0,parallel);
cumul_post(entries1,limit)
```



Machine Choice Cost Constraint

```
linear(sum);
forall(task2,machine_choice_task,
      sum += task2.cost_var
);
post(solver.cost == sum)
```

Enumeration

ChipFactory - Loaded from file D:\ChipFactory\Data\chemi.cod 2000/2/25 - 2000/3/3

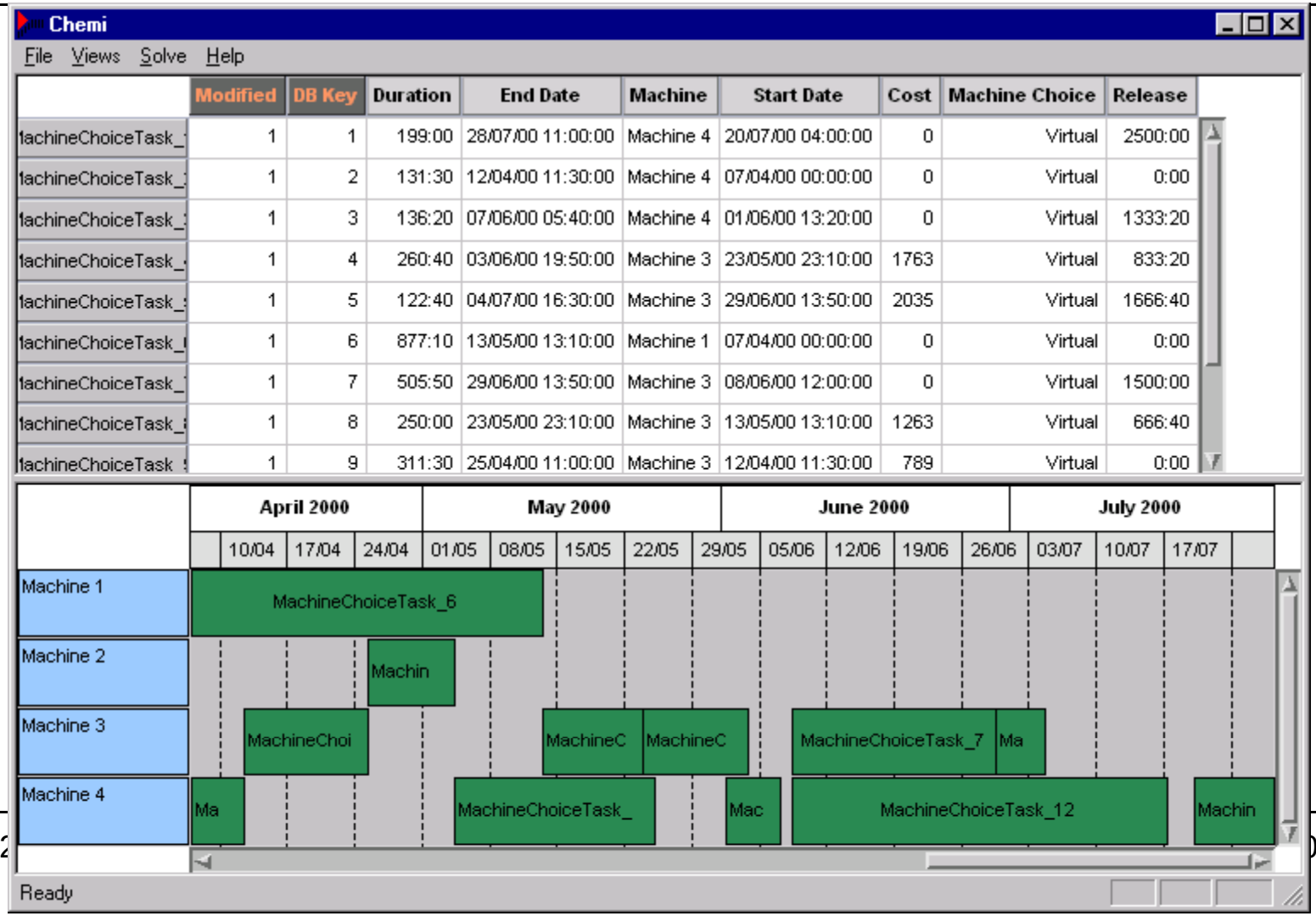
File Views Base Views Action Check Help

	Modified	Application_class	Nr	Description Enumeration	Type	Optimization	Partial Search	Solver
Enumeration_1	3	machine_choice_task	1	The labeling routine	Variable Choice	None	Complete	Solver for the

	Modified	Nr	Attribute	Description Enumeration_item	Method	First() Method	Last() Method	Next()
EnumerationItem_1	2	1	start_var	Assign start first	Min	unused	unused	
EnumerationItem_2	2	2	duration_var	Then try minimal duration	Min	unused	unused	
EnumerationItem_3	2	3	resource_var	At the end fix the resource	Min	unused	unused	
EnumerationItem_4	2	4	cost_var	At the end fix the cost	Min	unused	unused	

Ready

Finished Application



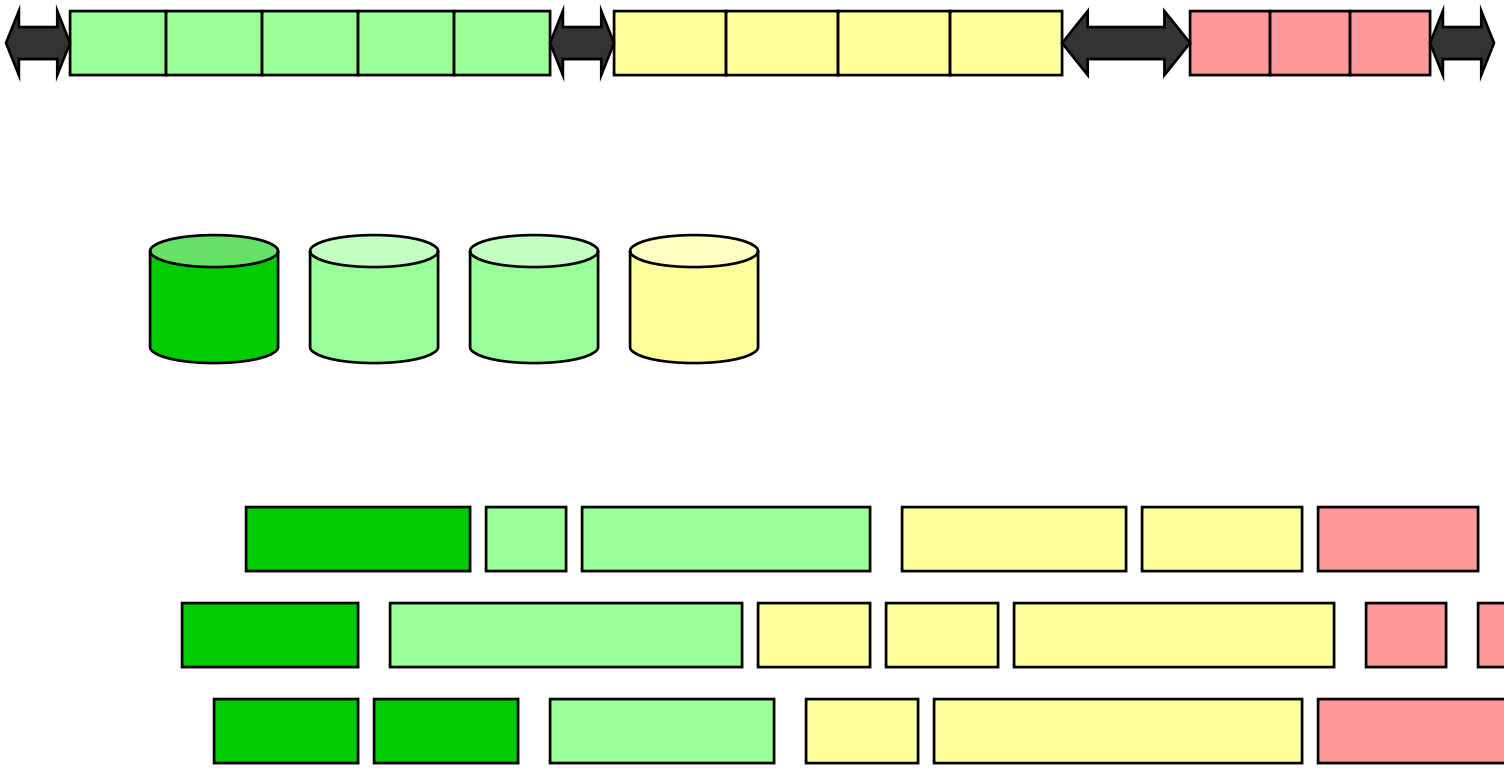
Example 3

Two-stage production process

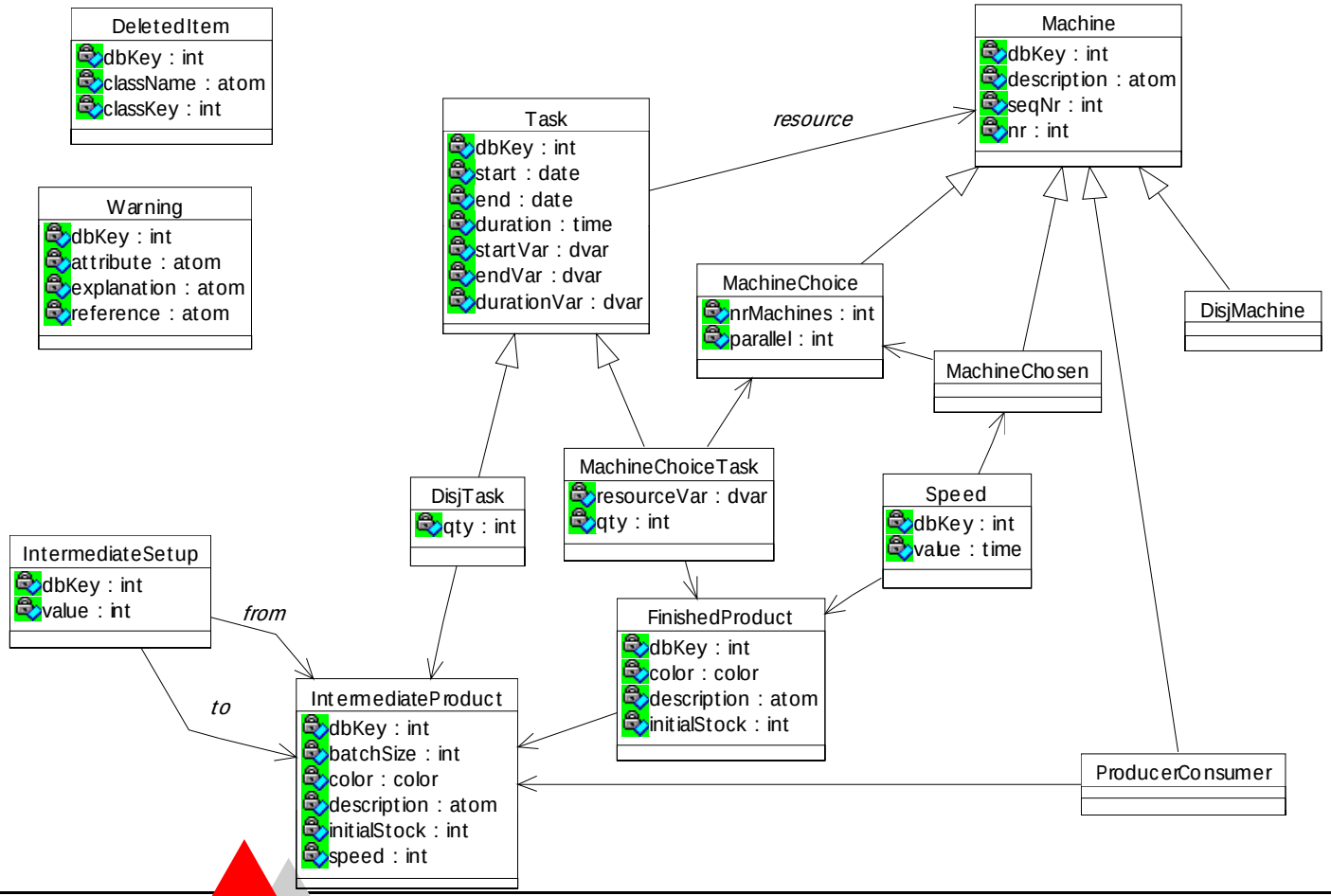
Problem Description

- **Example application from LISCOS Project (BASF)**
- **Two stage, continuous process scheduling problem**
- **First stage production**
  - **intermediate products (10)**
  - **batch based process**
  - **campaign based schedule**
  - **setup times/costs on one line**
- **Intermediate product storage**
  - **producer/consumer constraints**
- **Second stage production**
  - **finished products (100)**
  - **machine choice, machine restrictions**
  - **variable production rates on machines**

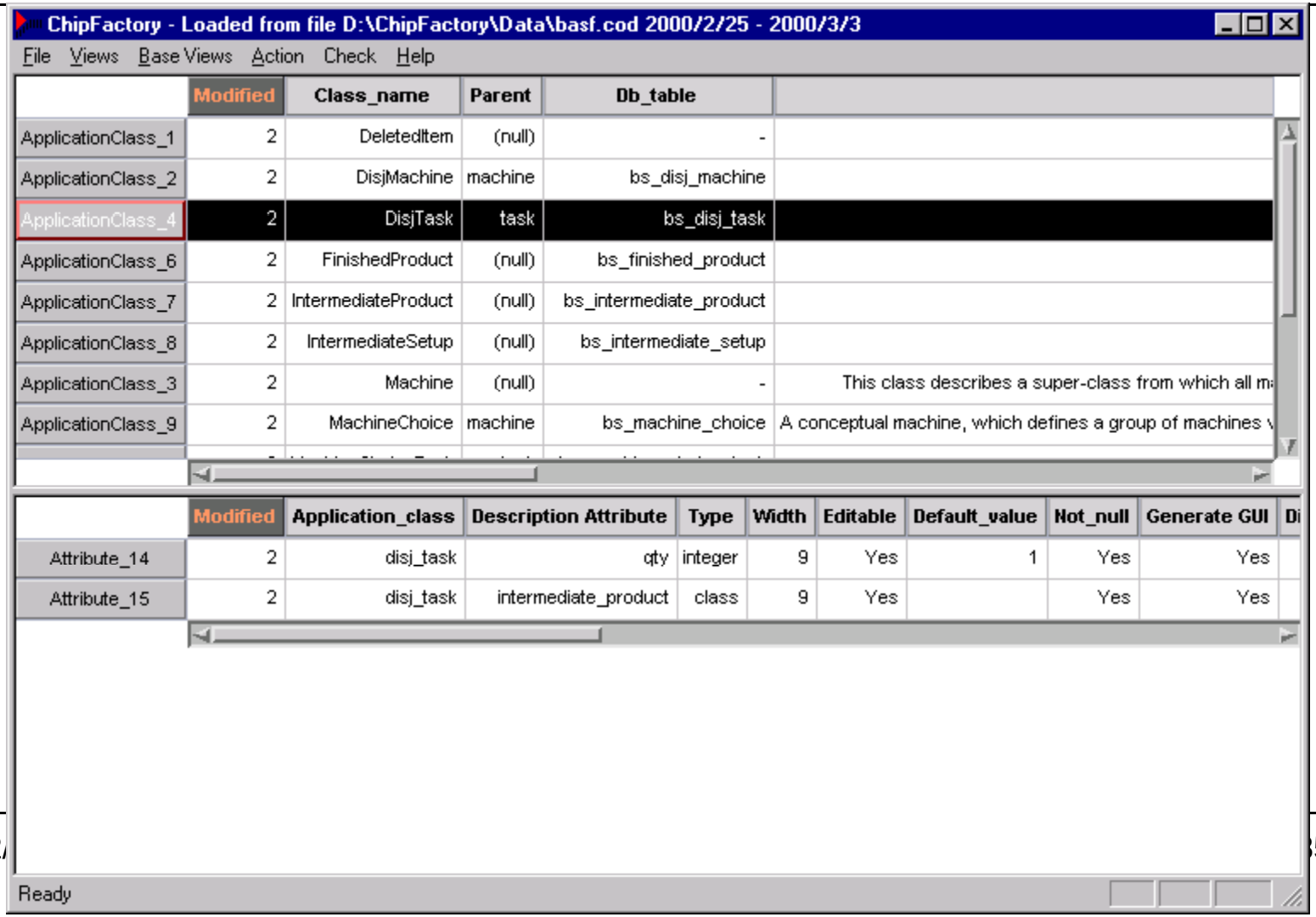
Problem Description (2)



Object Model



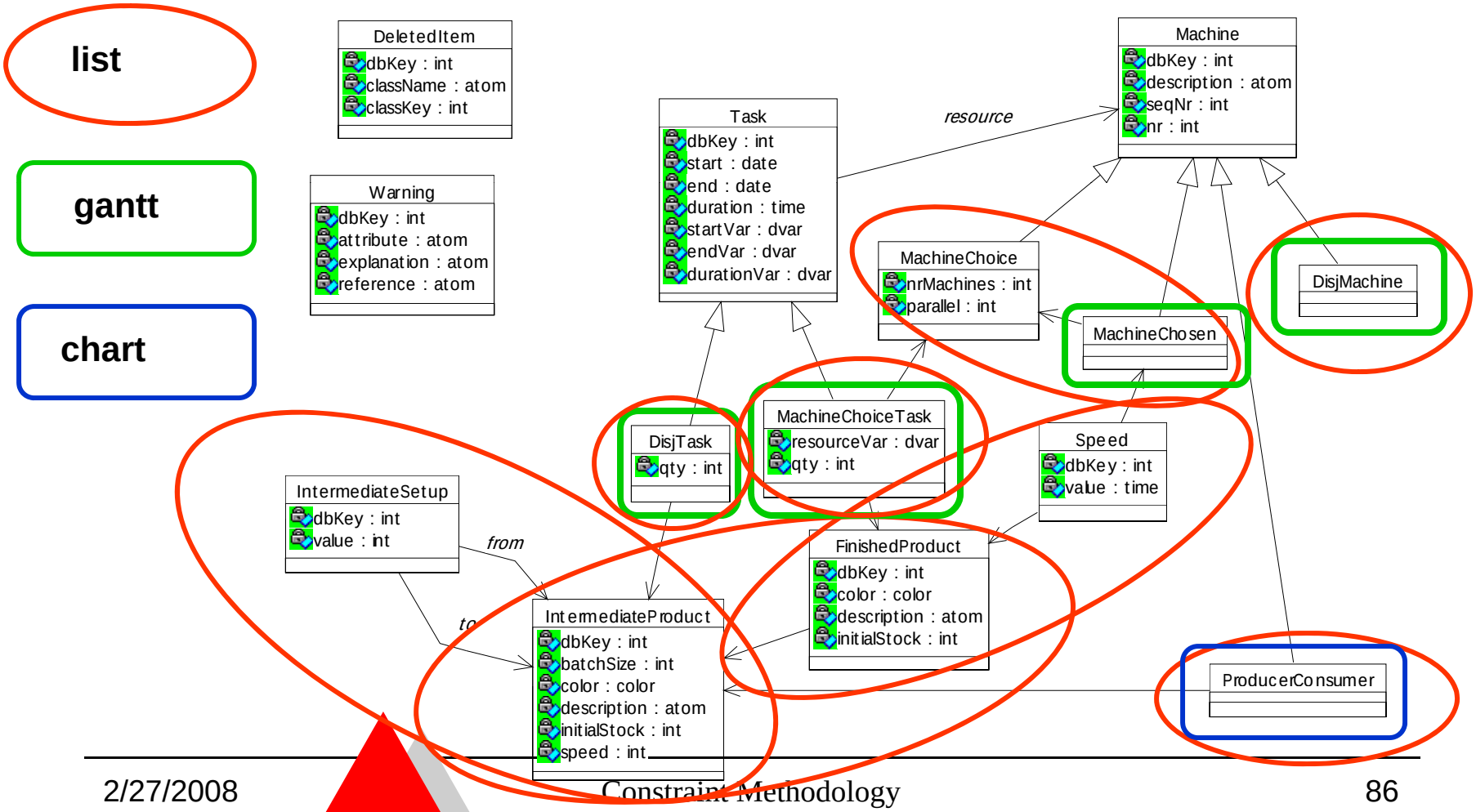
Object Definition



	Modified	Class_name	Parent	Db_table	
ApplicationClass_1	2	DeletedItem	(null)	-	
ApplicationClass_2	2	DisjMachine	machine	bs_disj_machine	
ApplicationClass_4	2	DisjTask	task	bs_disj_task	
ApplicationClass_6	2	FinishedProduct	(null)	bs_finished_product	
ApplicationClass_7	2	IntermediateProduct	(null)	bs_intermediate_product	
ApplicationClass_8	2	IntermediateSetup	(null)	bs_intermediate_setup	
ApplicationClass_3	2	Machine	(null)	-	This class describes a super-class from which all m
ApplicationClass_9	2	MachineChoice	machine	bs_machine_choice	A conceptual machine, which defines a group of machines

	Modified	Application_class	Description Attribute	Type	Width	Editable	Default_value	Not_null	Generate GUI	Di
Attribute_14	2	disj_task	qty	integer	9	Yes	1	Yes	Yes	
Attribute_15	2	disj_task	intermediate_product	class	9	Yes		Yes	Yes	

GUI Design


list
gantt
chart
DeletedItem
dbKey : int
className : atom
classKey : int
Warning
dbKey : int
attribute : atom
explanation : atom
reference : atom
Task
dbKey : int
start : date
end : date
duration : time
startVar : dvar
endVar : dvar
durationVar : dvar
resource
Machine
dbKey : int
description : atom
seqNr : int
nr : int
MachineChoice
nrMachines : int
parallel : int
MachineChosen
DisjMachine
DisjTask
qty : int
MachineChoiceTask
resourceVar : dvar
qty : int
Speed
dbKey : int
value : time
IntermediateSetup
dbKey : int
value : int
from
to
FinishedProduct
dbKey : int
color : color
description : atom
initialStock : int
IntermediateProduct
dbKey : int
batchSize : int
color : color
description : atom
initialStock : int
speed : int
ProducerConsumer

GUI Description

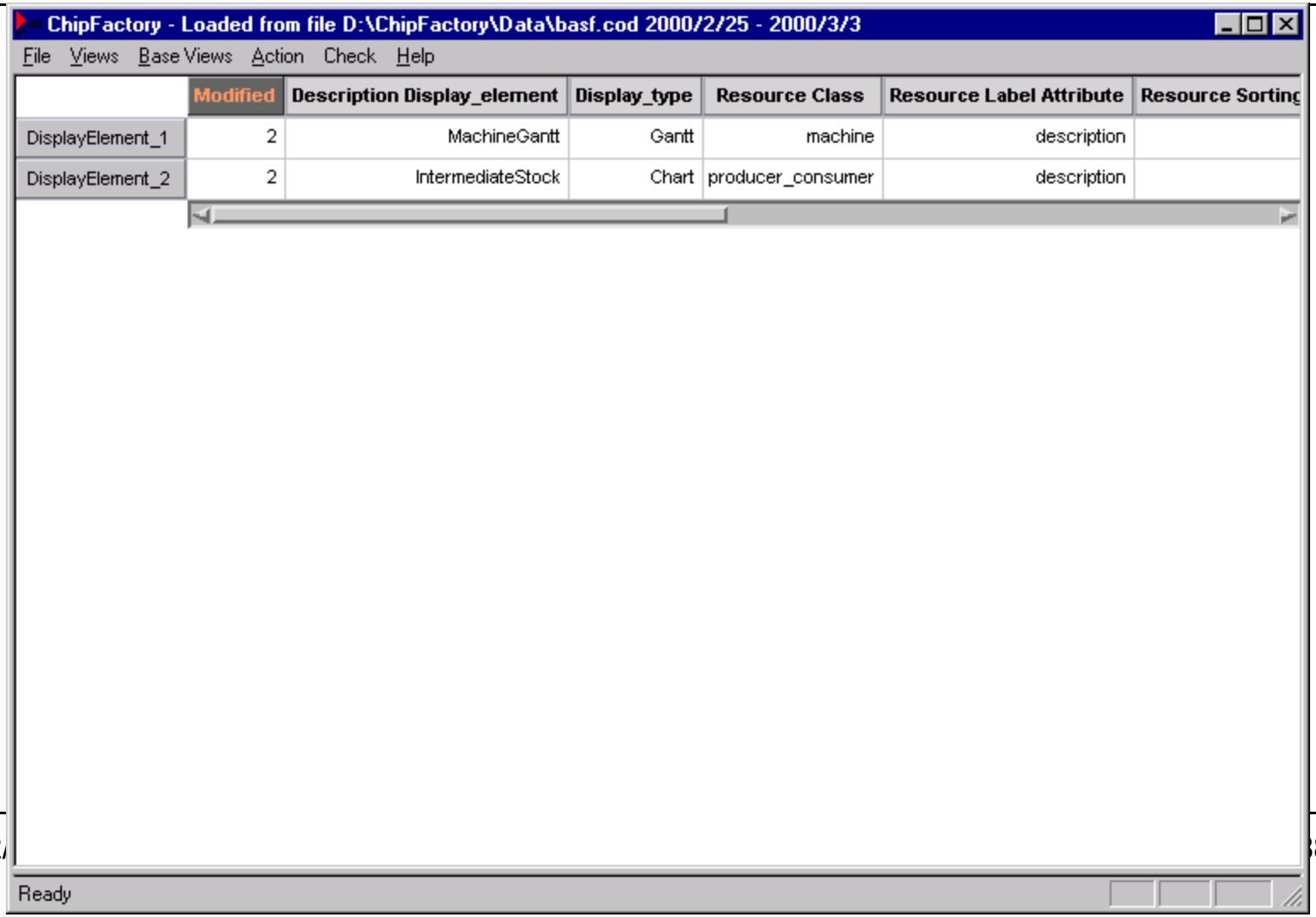
ChipFactory - Loaded from file D:\ChipFactory\Data\basf.cod 2000/2/25 - 2000/3/3

File Views Base Views Action Check Help

	Modified	Label	Callback	Application_class Top	Menu	Nr	Display_type	Known Cal
View_1	2	MachineChoice	OnViewMachineChoice	machine_choice	view	8	List x2	
View_2	2	Speed	OnViewSpeed	finished_product	view	9	List x2	
View_3	2	IntermediateProduct	OnViewIntermediateProduct	intermediate_product	view	10	List x2	
View_4	2	Setup	OnViewSetup	intermediate_product	view	11	List x2	
View_5	2	MachineGantt	OnViewMachineGantt	machine_choice_task	baseview	12	Gantt	
View_6	2	IntermediateStock	OnViewIntermediateStock	producer_consumer	baseview	13	Chart	
View_7	2	DisjMachine	OnViewDisjMachine	disj_machine	baseview	14	List	
View_8	2	ProducerConsumer	OnViewProducerConsumer	producer_consumer	baseview	15	List	
View_9	2	DisjTask	OnViewDisjTask	disj_task	baseview	16	List	
View_10	2	MachineChoiceTask	OnViewMachineChoiceTask	machine_choice_task	baseview	17	List	

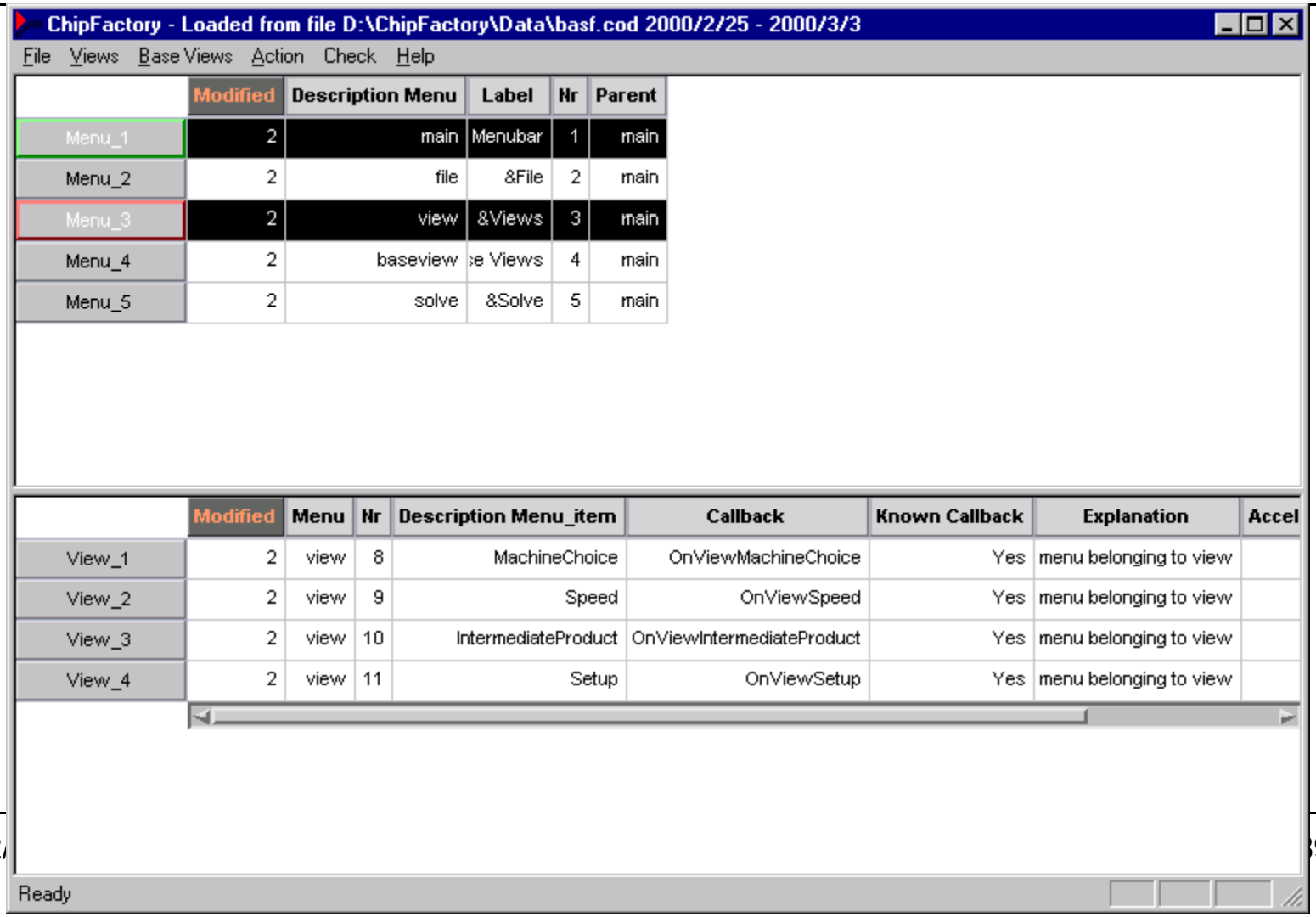
Ready

Display Elements



	Modified	Description Display_element	Display_type	Resource Class	Resource Label Attribute	Resource Sorting
DisplayElement_1	2	MachineGantt	Gantt	machine	description	
DisplayElement_2	2	IntermediateStock	Chart	producer_consumer	description	

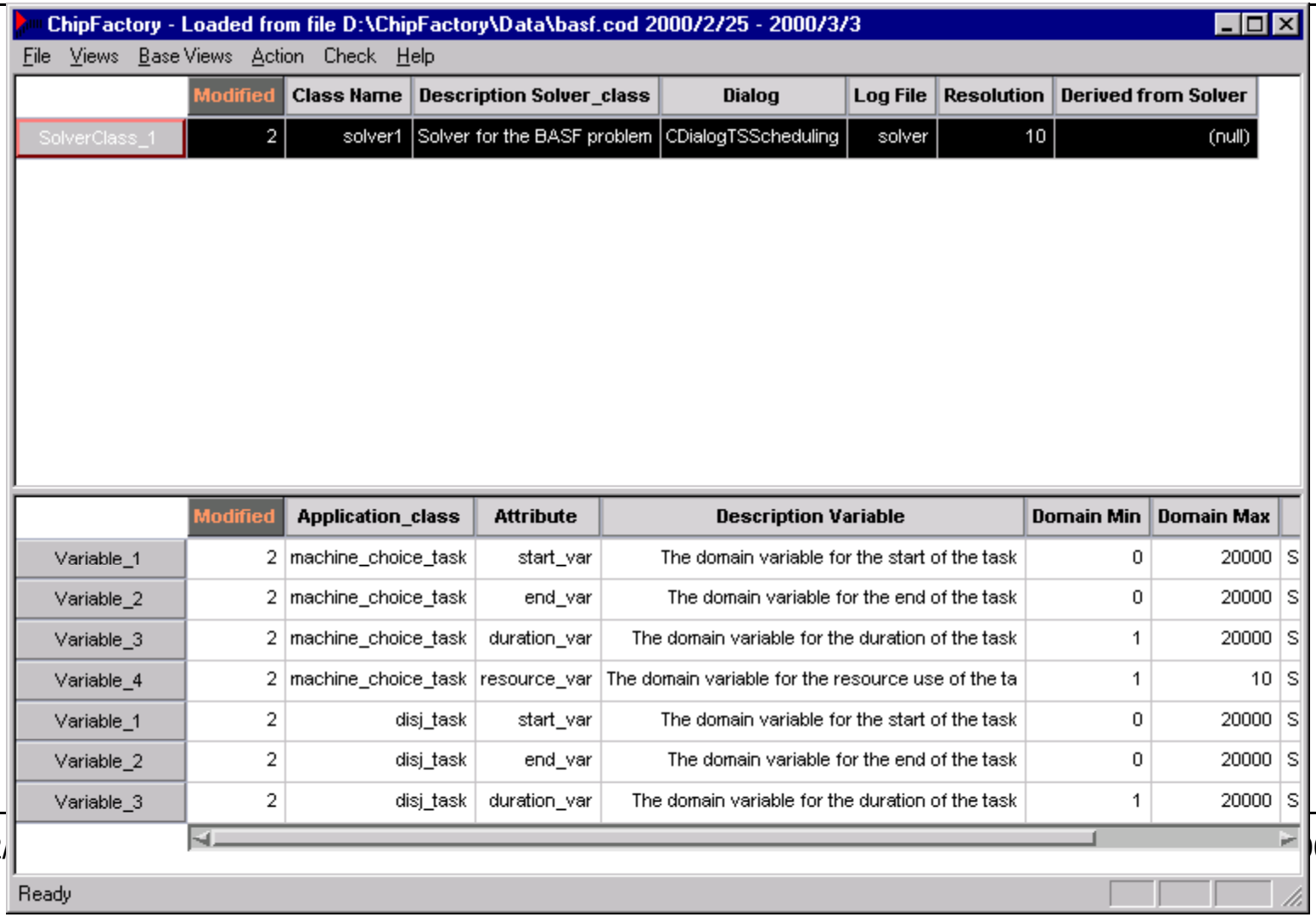
Menu Description



	Modified	Description Menu	Label	Nr	Parent
Menu_1	2	main	Menubar	1	main
Menu_2	2	file	&File	2	main
Menu_3	2	view	&Views	3	main
Menu_4	2	baseview	se Views	4	main
Menu_5	2	solve	&Solve	5	main

	Modified	Menu	Nr	Description Menu_item	Callback	Known Callback	Explanation	Accel
View_1	2	view	8	MachineChoice	OnViewMachineChoice	Yes	menu belonging to view	
View_2	2	view	9	Speed	OnViewSpeed	Yes	menu belonging to view	
View_3	2	view	10	IntermediateProduct	OnViewIntermediateProduct	Yes	menu belonging to view	
View_4	2	view	11	Setup	OnViewSetup	Yes	menu belonging to view	

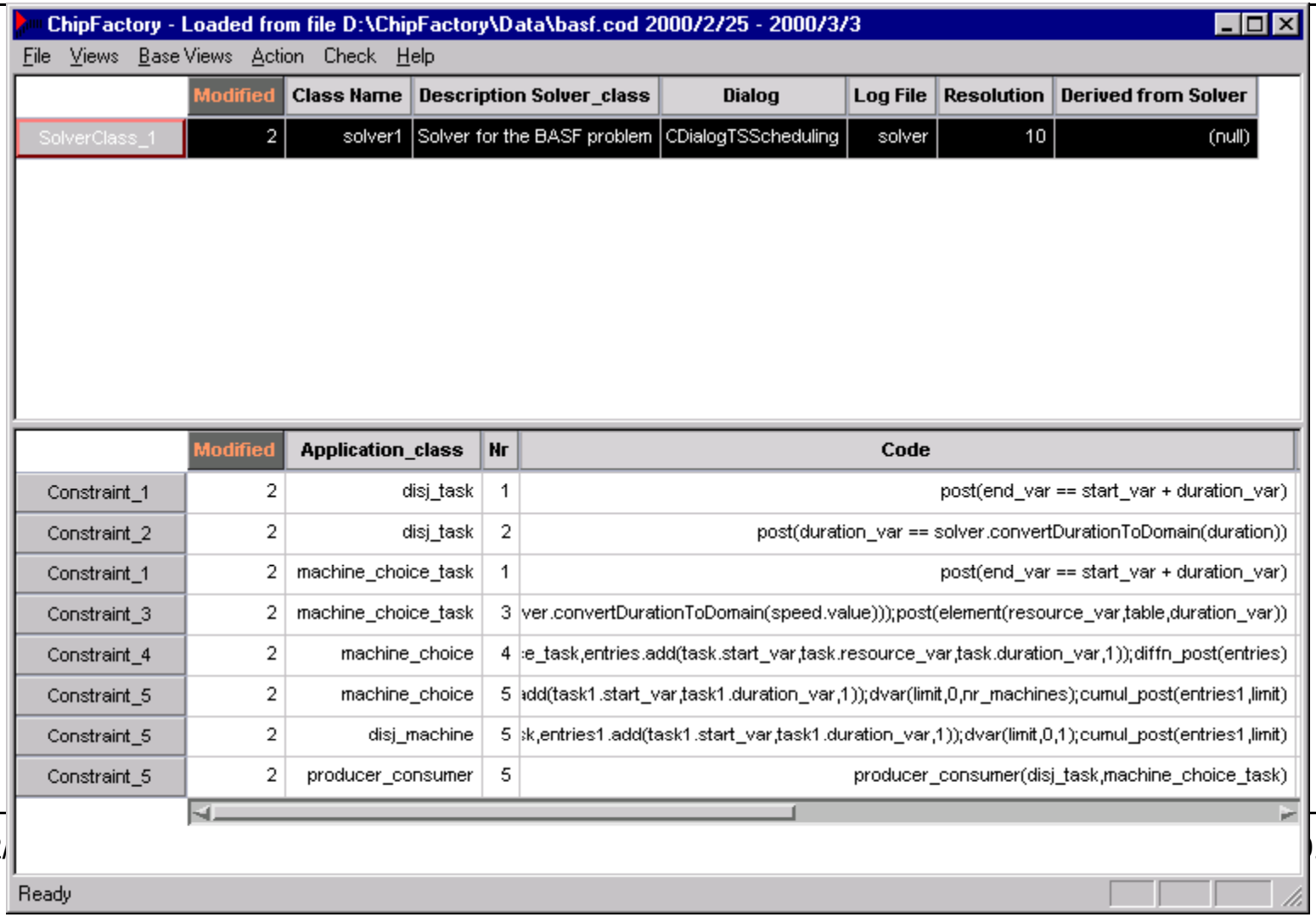
Variable Description



	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the BASF problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Attribute	Description Variable	Domain Min	Domain Max	
Variable_1	2	machine_choice_task	start_var	The domain variable for the start of the task	0	20000	S
Variable_2	2	machine_choice_task	end_var	The domain variable for the end of the task	0	20000	S
Variable_3	2	machine_choice_task	duration_var	The domain variable for the duration of the task	1	20000	S
Variable_4	2	machine_choice_task	resource_var	The domain variable for the resource use of the ta	1	10	S
Variable_1	2	disj_task	start_var	The domain variable for the start of the task	0	20000	S
Variable_2	2	disj_task	end_var	The domain variable for the end of the task	0	20000	S
Variable_3	2	disj_task	duration_var	The domain variable for the duration of the task	1	20000	S

Constraint Description

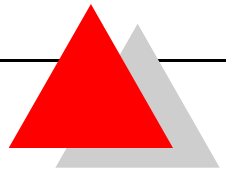


	Modified	Class Name	Description Solver_class	Dialog	Log File	Resolution	Derived from Solver
SolverClass_1	2	solver1	Solver for the BASF problem	CDialogTSScheduling	solver	10	(null)

	Modified	Application_class	Nr	Code
Constraint_1	2	disj_task	1	post(end_var == start_var + duration_var)
Constraint_2	2	disj_task	2	post(duration_var == solver.convertDurationToDomain(duration))
Constraint_1	2	machine_choice_task	1	post(end_var == start_var + duration_var)
Constraint_3	2	machine_choice_task	3	ver.convertDurationToDomain(speed.value)));post(element(resource_var,table,duration_var))
Constraint_4	2	machine_choice	4	e_task,entries.add(task.start_var,task.resource_var,task.duration_var,1));diffn_post(entries)
Constraint_5	2	machine_choice	5	dd(task1.start_var,task1.duration_var,1));dvar(limit,0,nr_machines);cumul_post(entries1,limit)
Constraint_5	2	disj_machine	5	k,entries1.add(task1.start_var,task1.duration_var,1));dvar(limit,0,1);cumul_post(entries1,limit)
Constraint_5	2	producer_consumer	5	producer_consumer(disj_task,machine_choice_task)

Constraint Model

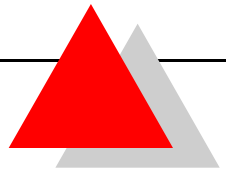
- **DisjTask constraints**
- **MachineChoiceTask constraints**
- **DisjMachine resource constraint**
- **Machine choice resource constraint**
- **Producer/Consumer constraint**



DisjTask constraints

post(end_var == start_var + duration_var)

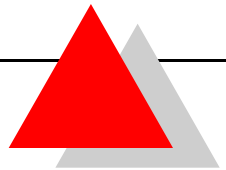
post(duration_var == solver.convertDurationToDomain(duration))



MachineChoiceTask constraints

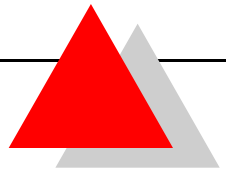
```
post(end_var == start_var + duration_var)

int(table,machine_choice.nr_machines);
forall(speed,speed,
    if(speed.finished_product == finished_product,
        table(speed.machine_chosen.nr) =
            qty *solver.convertDurationToDomain(speed.value)
    )
);
post(element(resource_var,table,duration_var))
```



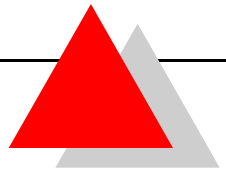
DisjMachine constraint

```
cumul_entries(entries1);
forall(task1,disj_task,
      entries1.add(task1.start_var,task1.duration_var,1)
);
dvar(limit,0,1);
cumul_post(entries1,limit)
```



MachineChoice constraint

```
diffn_entries(entries);
forall(task,machine_choice_task,
      entries.add(task.start_var,task.resource_var,task.duration_var,1)
);
diffn_post(entries)
cumul_entries(entries1);
forall(task1,machine_choice_task,
      entries1.add(task1.start_var,task1.duration_var,1)
);
dvar(limit,0,nr_machines);
cumul_post(entries1,limit)
```

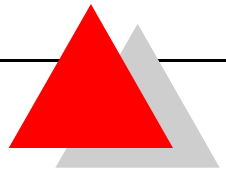


Producer/Consumer constraint

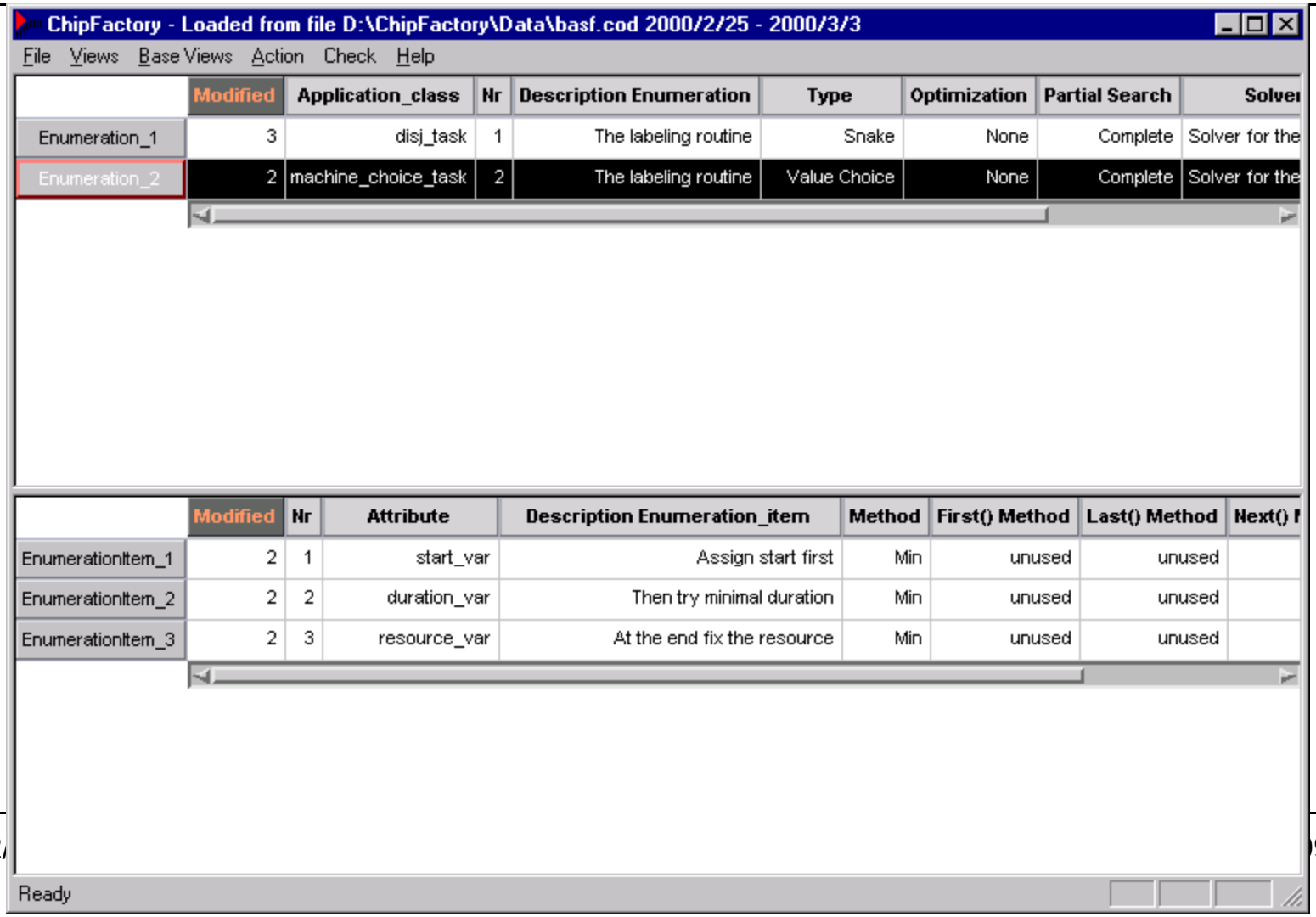
```
int(sum);
cumul_entries(entries);
forall(p,Producer,
      if (p.intermediate_product == intermediate_product) then {
            entries.add(0,p.end_var,p.qty,p.end_var);
            sum += p.qty
      }
);
sum += intermediate_product.initial_stock;
```

Producer/Consumer constraint (2)

```
forall(c,Consumer,
    if (c.finished_product.intermediate_product == intermediate_product) then {
        dvar(duration_temp,0,20000);
        post(c.start_var + duration_temp == 20000);
        entries.add(c.start_var,duration_temp,c.qty,20000)
    }
);
dvar(limit,0,sum);
dvar(end,0,20000);
cumul_post(entries,limit,end)
```



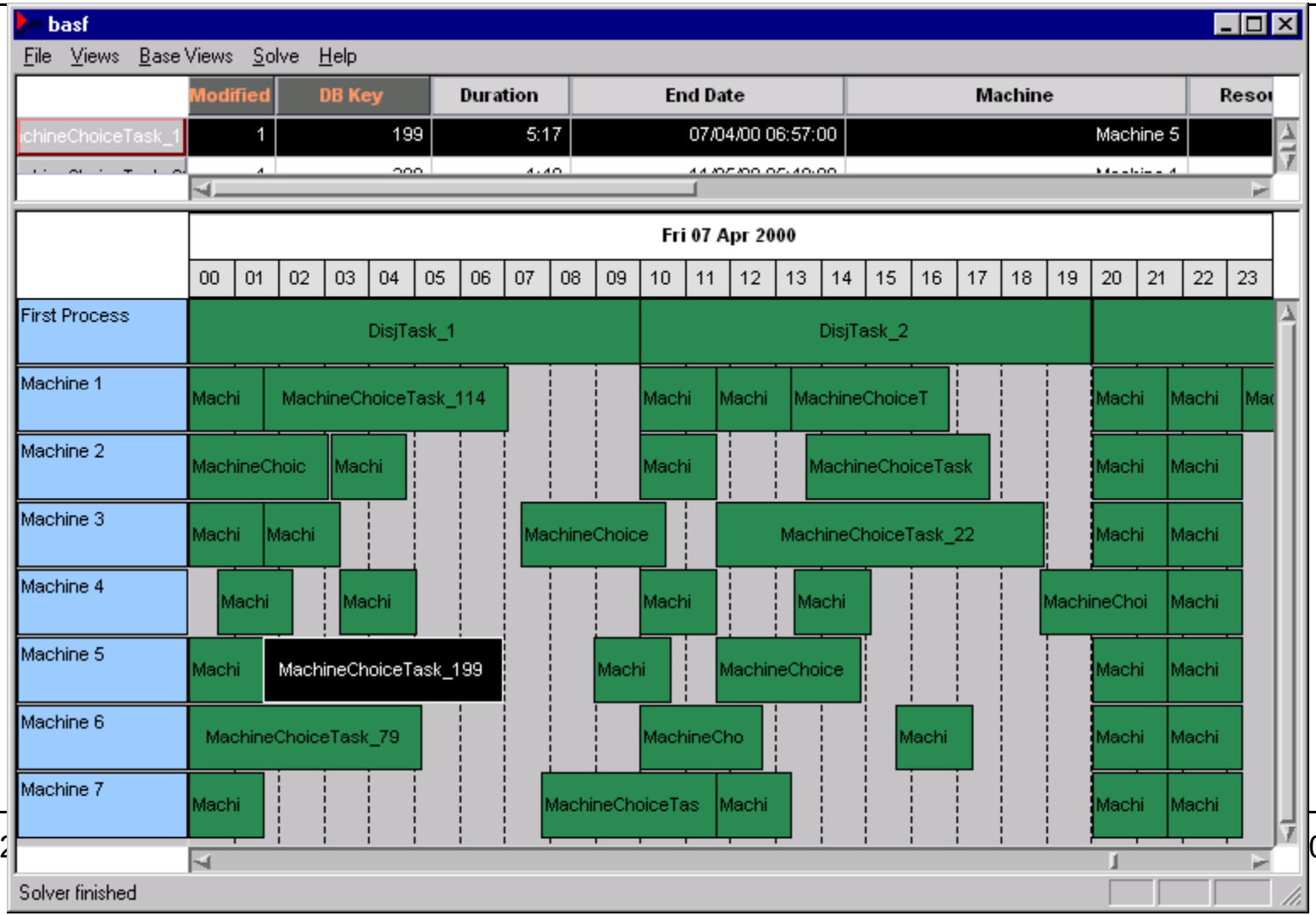
Enumeration



	Modified	Application_class	Nr	Description Enumeration	Type	Optimization	Partial Search	Solver
Enumeration_1	3	disj_task	1	The labeling routine	Snake	None	Complete	Solver for the
Enumeration_2	2	machine_choice_task	2	The labeling routine	Value Choice	None	Complete	Solver for the

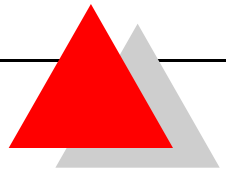
	Modified	Nr	Attribute	Description Enumeration_item	Method	First() Method	Last() Method	Next()
EnumerationItem_1	2	1	start_var	Assign start first	Min	unused	unused	
EnumerationItem_2	2	2	duration_var	Then try minimal duration	Min	unused	unused	
EnumerationItem_3	2	3	resource_var	At the end fix the resource	Min	unused	unused	

Finished Application



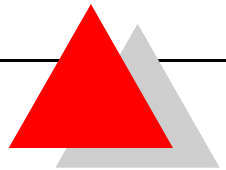
Part III

30 Golden Rules for CP Modeling

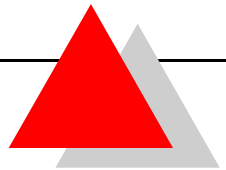


Caveats

- **Rules roughly grouped in 5 categories**
  - **variables**
  - **constraints**
  - **search**
  - **data**
  - **process**
- **This works for me**



Variables

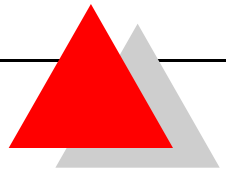


Rule 1

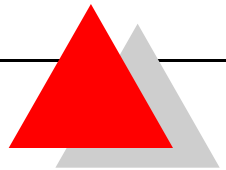
- **Beware of unknown number of activities**
  - **“The system should decide if there is one or two runs of this product”**
  - **variables must be defined in the beginning**
  - **can not express constraints over unknown objects**
  - **decompose into planning and scheduling**

Rule 2

- **Better few variables with large domains than many variables with small domains**
  - **really look for amount of propagation possible**
- **0/1 variables mean something's wrong**
  - **no propagation**
  - **no heuristic**
  - **alternative model will use global constraints**

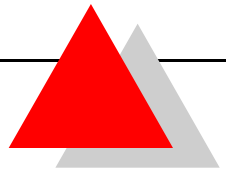


Constraints



Rule 3

- **Use the existing global constraints**
  - **>> Besser schlecht gefahren als gut gelaufen <<**
  - **Clearly depends on the system**
  - **CHR users would see that differently**



Rule 4

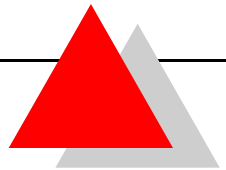
- **Dynamic dependencies between several activities**
  - **“speed/duration depends on previous tasks”**
  - **“tasks belong to the same shift if there is no gap of more than 4 hours, this depends on many other tasks”**
  - **problems with lower bounds/feasibility analysis**
  - **very little propagation until the end is reached**
  - **left to right scheduling may help**

Rule 5

- **Beware of conditional constraints**
  - **“schedule as many orders as you can”**
  - **possible to cancel orders/tasks**
  - **rather weak propagation**
  - **careful when introducing dummy tasks to handle uncertainty**

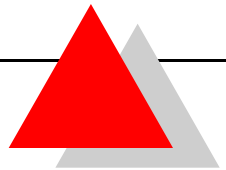
Rule 6

- **Write a constraint checker first**
  - **“yes, this is a hard constraint”**
  - **test ability to express constraints at all**
  - **check the existing solution and explain inconsistencies**
  - **surprising how many constraints are not constraints, but are preferences**



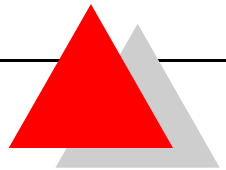
Rule 7

- **Too many soft constraints**
  - **“when the system can not find a solution, it should remove some of these constraints”**
  - **no propagation until all softness disappears**
  - **problem may be better suited to local search**



Rule 8

- **Model consists of inequalities only**
  - **not problem, but model specific**
  - **re-model using global constraints**
  - **change to other solver, IP tools**



Rule 9

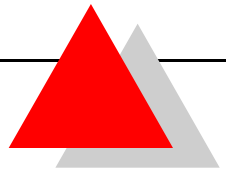
- **Constraints like complexity**
  - **the more the merrier**
  - **model 100% of problem, nobody else can do this**
  - **if a feasible solution is easy, constraints may not be the right tool**

Rule 10

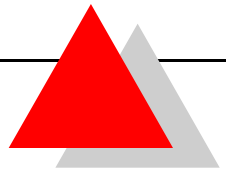
- **Go for first principles**
  - **understand the reason behind the constraint**
  - **remodel if possible to simplify**

Rule 11

- **Don’t write your own constraints**
  - **do not mix constraints & application development at the same time**
  - **test on known problems before use**



Search



Rule 12

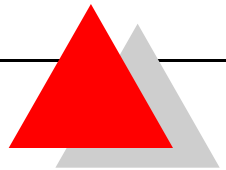
- **Think of an escape route**
  - **it should always be possible to find a feasible solution**
  - **turn off constraints until a solution is possible**
  - **option to release due dates or release dates**
  - **humans can violate the constraints, the system should not**

Rule 13

- **Trouble with reactive rescheduling**
  - **“and while I move this task in the Gantt chart, it should make sure the solution stays consistent”**
  - **a small change can have a huge impact**
  - **it can take a long time to set up all constraints again**

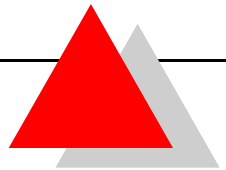
Rule 14

- **Careful with cost**
  - **“and the system should find the best compromise among these costs”**
  - **avoid additive cost when optimizing**
  - **just because the user wants to see a particular cost value in the end does not mean that you have to use this inside the solver**
  - **use partial search to explore more of the search space**
  - **impose hard limits on different cost dimensions**
  - **use heuristics to find good solution**

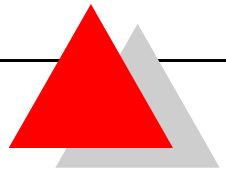


Rule 15

- **Visualize early, visualize often**
  - **so many data, so little time**
  - **draw Gantt charts, diagrams etc**
  - **incremental display shows problem areas in search**
  - **often requested by end-users when shown during debugging**
  - **pin your object model diagram to the wall**



Data

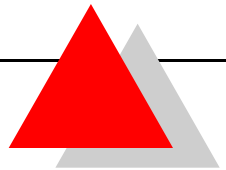


Rule 16

- **Do as I do, don't do as I say**
  - **check actual figures, do not trust explanations**
  - **always ask for an existing solution right from the start**

Rule 17

- **Take small data set first, then use real data**
  - **hand check results at least once**
  - **understand small problem completely**
  - **see if you (or the user) can improve the result**

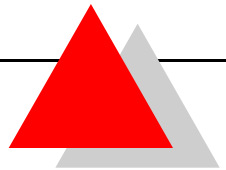


Rule 18

- **Don’t get hooked on a single data set**
  - **testing on one data set means that this will be the only working data set**
  - **for daily schedule require at least 6 month of data**
  - **look for seasonal variations**

Rule 19

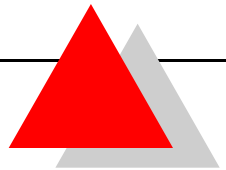
- **Check your data**
  - **often: last problems only sorted out in acceptance testing**
  - **write dedicated data checking**
    - **even if it looks like a waste of time**
    - **CHIP Factory generates syntactic checks for you**
  - **trust nobody (neither machine nor persons)**
  - **be paranoid (they *are* out there to get you)**



Process

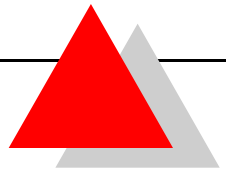
Rule 20

- **One thing at a time**
  - **do not try to put everything in at the same time**
  - **classical rule: change one parameter of an experiment at a time**



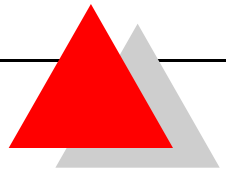
Rule 21

- **Do not believe other people's model**
  - **in particular: OR people**
  - **models are often linked to features of solver**



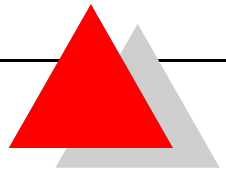
Rule 22

- **Get it right, then get it fast**
  - **no need to optimize until all constraints are taken into account**
  - **optimize on real data only**
  - **more propagation is better than fast setup of constraint**
  - **always worry about performance in data preparation**



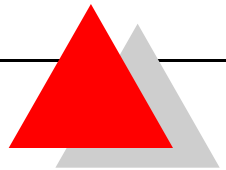
Rule 23

- **Beware of the bright new idea**
  - **projects introducing new business processes often fail because of the change, not because of the solver**
  - **less risk in automating existing process**
  - **no risk, no fun**



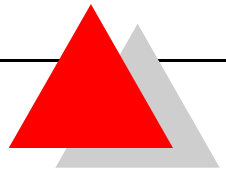
Rule 24

- **Look for the dual problem formulation**
  - **reverse roles of activities and resources**
  - **look for predecessor instead of successor**
  - **exchange start and end in schedule**
  - **possible to combine both formulations in one model**



Rule 25

- **Use those experts**
  - **domain knowledge makes the difference**
  - **always talk to the end-users**
    - **not management**
    - **not IT people**
    - **not consultants**
  - **emulates what is currently done (+-)**
  - **problem: making the expert redundant**



Rule 26

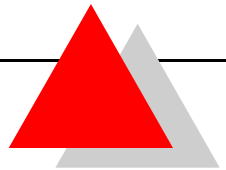
- **Get it right the first time**
  - **don’t rush through design**
  - **in particular data model, database**
  - **wrong structure impossible to fix**
  - **easy to add constraints, new attributes**
  - **do prototypes to know what we are talking about**

Rule 27

- **Explain the problem to the expert and to an outsider**
  - **avoid expensive misunderstandings**
  - **get fresh ideas**

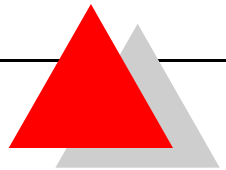
Rule 28

- **If it doesn't work, then it doesn't work**
  - **cheaper to redesign than to endlessly fix wrong design**
  - **there is always a possibility to define the problem to suit your method**



Rule 29

- **Is there a better way?**
  - **reserve time for experiments**
  - **try at least one odd idea**
  - **compare to others, even on their terms**



Rule 30

- **A prototype is worth a thousand words**
  - **quick and dirty (manual data preparation)**
  - **be prepared to throw it away**

Conclusions

- **Modeling is engineering rather than science**
  - **perhaps art, not engineering**
- **Right tools can help to speed up process**
  - **CHIP factory allows very rapid development**
  - **iterative design gives much superior end-product**
- **Problem solver only part of project**
  - **users want solutions, not technology**
- **If it was simple, they wouldn't need us**
  - **problems inherently complex**
  - **technology very powerful**