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
Background

◆ **CHIC ESPRIT Project**
  - A. Chamard, B. Fischer, B. Guinaudeau, A. Guillard: 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, Edinburgh, 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

2/27/2008
Principles of Operation

Specification

- C++ code
- Database generation
- Documentation
- Test routines
- GUI
- Solver
- File I/O

Custom
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

- **DeletedItem**
  - `dbKey`: int
  - `className`: atom
  - `classKey`: int

- **Warning**
  - `dbKey`: int
  - `attribute`: atom
  - `explanation`: atom
  - `reference`: atom

- **IntermediateSetup**
  - `dbKey`: int
  - `value`: int

- **IntermediateProduct**
  - `dbKey`: int
  - `batchSize`: int
  - `color`: color
  - `description`: atom
  - `initialStock`: int
  - `speed`: int

- **Task**
  - `dbKey`: int
  - `start`: date
  - `end`: date
  - `duration`: time
  - `startVar`: dvar
  - `endVar`: dvar
  - `durationVar`: dvar

- **Machine**
  - `dbKey`: int
  - `description`: atom
  - `seqNr`: int
  - `nrr`: int

- **DisjMachine**

- **DisjTask**
  - `qty`: int

- **MachineChoice**
  - `nrMachines`: int
  - `parallel`: int

- **MachineChoiceTask**
  - `resourceVar`: dvar
  - `qty`: int

- **Speed**
  - `dbKey`: int
  - `value`: time

- **FinishedProduct**
  - `dbKey`: int
  - `color`: color
  - `description`: atom
  - `initialStock`: int
  - `speed`: int

- **ProducerConsumer**

- **IntermediateSetup**
  - `from`

- **IntermediateProduct**
  - `to`
Classes

- Described by class name, attributes and links

Class name

Attributes (Name, Type)

Methods
(automatically derived)
Derivation

- New classes can be derived from existing ones

```
Task
  dbKey : int
  start : date
  end : date
  duration : time
  startVar : dvar
  endVar : dvar
  durationVar : dvar

DisjTask
  qty : int

MachineChoiceTask
  resourceVar : dvar
  qty : int
```
Associations, Navigation, Aggregation

- Links between classes, which allow navigation between them

![Diagram of class links](image-url)
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
<table>
<thead>
<tr>
<th>Modified</th>
<th>DB Key</th>
<th>Description</th>
<th>Hr</th>
<th>Seq Hr</th>
<th>Intermediate Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>P/C 1</td>
<td>0</td>
<td>1</td>
<td>IP 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>P/C 2</td>
<td>0</td>
<td>1</td>
<td>IP 2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>P/C 3</td>
<td>0</td>
<td>1</td>
<td>IP 3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>P/C 4</td>
<td>0</td>
<td>1</td>
<td>IP 4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>P/C 5</td>
<td>0</td>
<td>1</td>
<td>IP 5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>P/C 6</td>
<td>0</td>
<td>1</td>
<td>IP 6</td>
</tr>
</tbody>
</table>
### Double List

<table>
<thead>
<tr>
<th>Modified</th>
<th>DB Key</th>
<th>Description</th>
<th>Iter</th>
<th>Seq Iter</th>
<th>Iter Machines</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineChosen_1</td>
<td>1</td>
<td>Machine 1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_2</td>
<td>1</td>
<td>Machine 2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_3</td>
<td>1</td>
<td>Machine 3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_4</td>
<td>1</td>
<td>Machine 4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_5</td>
<td>1</td>
<td>Machine 5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_6</td>
<td>1</td>
<td>Machine 6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MachineChosen_7</td>
<td>1</td>
<td>Machine 7</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Chart

<table>
<thead>
<tr>
<th>Modified</th>
<th>DB Key</th>
<th>Description</th>
<th>Seq Nr</th>
<th>Resources Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>CumulMachine_1</td>
<td>1</td>
<td>CumulMachine1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fri 07 Apr 2000**

![Chart Image]
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

<table>
<thead>
<tr>
<th>Task nr</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Resource Use</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task nr</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Resource Use</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- 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

Task
- dbKey : int
- start : date
- end : date
- duration : time
- startVar : dvar
- endVar : dvar
- durationVar : dvar

Machine
- dbKey : int
- description : atom
- seqNr : int

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
## Project Definition

### Chip Factory - Loaded from file D:\ChipFactory\Data\ship.cod 2000/2/25 - 2000/3/3

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application Name</th>
<th>Directory</th>
<th>Initial View</th>
<th>Attr Prefix</th>
<th>Database</th>
<th>Description Project</th>
<th>Executable name</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project_1</td>
<td>2</td>
<td>Ship</td>
<td>D:\ship</td>
<td>CumulChart</td>
<td>ship</td>
<td>SHP</td>
<td>ship</td>
<td></td>
</tr>
</tbody>
</table>

[Image of a table with columns for Modified, Application Name, Directory, Initial View, Attr Prefix, Database, Description Project, Executable name, and File, along with a text box indicating the project details.]
# Class Definition

## Table 1: ApplicationClass

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class_name</th>
<th>Parent</th>
<th>Db_table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ApplicationClass_1</td>
<td>CumulMachine</td>
<td>machine</td>
<td>shp_cumul_machine</td>
</tr>
<tr>
<td>2</td>
<td>ApplicationClass_2</td>
<td>CumulTask</td>
<td>task</td>
<td>shp_cumul_task</td>
</tr>
<tr>
<td>2</td>
<td>ApplicationClass_3</td>
<td>DeletedItem</td>
<td>(null)</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>ApplicationClass_4</td>
<td>Machine</td>
<td>(null)</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>ApplicationClass_5</td>
<td>Precedence</td>
<td>(null)</td>
<td>shp_precedence</td>
</tr>
</tbody>
</table>

## Table 2: Attribute

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Description</th>
<th>Attribute</th>
<th>Type</th>
<th>Width</th>
<th>Editable</th>
<th>Default_value</th>
<th>Not_null</th>
<th>Generate GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>cumul_task</td>
<td>resource_required</td>
<td>integer</td>
<td>9</td>
<td>Yes</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>cumul_task</td>
<td>resource_used</td>
<td>integer</td>
<td>9</td>
<td>No</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>cumul_task</td>
<td>resource_used_var</td>
<td>dvar</td>
<td>0</td>
<td>No</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## GUI Definition

<table>
<thead>
<tr>
<th>Modified</th>
<th>Label</th>
<th>Callback</th>
<th>Application_class Top</th>
<th>Menu</th>
<th>Nr</th>
<th>Display_type</th>
<th>Known Callback</th>
<th>Accel</th>
</tr>
</thead>
<tbody>
<tr>
<td>View_1</td>
<td>CumulTask</td>
<td>OnViewCumulTask</td>
<td>cumul_task</td>
<td>view</td>
<td>8</td>
<td>List</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View_2</td>
<td>CumulMachine</td>
<td>OnViewCumulMachine</td>
<td>cumul_machine</td>
<td>view</td>
<td>9</td>
<td>List</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View_3</td>
<td>Precedence</td>
<td>OnViewPrecedence</td>
<td>precedence</td>
<td>view</td>
<td>10</td>
<td>List</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View_4</td>
<td>CumulChart</td>
<td>OnViewCumulChart</td>
<td>cumul_machine</td>
<td></td>
<td>11</td>
<td>Chart</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
## Display Elements

<table>
<thead>
<tr>
<th>Modified</th>
<th>Description</th>
<th>Display_element</th>
<th>Display_type</th>
<th>Resource Class</th>
<th>Resource Label Attribute</th>
<th>Resource Sorting #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DisplayElement_1</td>
<td>CumulChart</td>
<td>Chart</td>
<td>cumul_machine</td>
<td>name</td>
</tr>
</tbody>
</table>
## Menu Definition

### Menu Structure

<table>
<thead>
<tr>
<th>Modified</th>
<th>Description Menu</th>
<th>Label</th>
<th>Nr</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Menu_1</td>
<td>main</td>
<td>1</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>Menu_2</td>
<td>file</td>
<td>2</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td>Menu_3</td>
<td>view</td>
<td>3</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>Menu_4</td>
<td>solve</td>
<td>4</td>
<td>main</td>
</tr>
</tbody>
</table>

### Menu Items

<table>
<thead>
<tr>
<th>Modified</th>
<th>Menu Item</th>
<th>Nr</th>
<th>Description Menu Item</th>
<th>Callback</th>
<th>Known Callback</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MenuItem_1</td>
<td>1</td>
<td>file</td>
<td>New</td>
<td>onInstance</td>
<td>Create sample data</td>
</tr>
<tr>
<td>2</td>
<td>MenuItem_2</td>
<td>2</td>
<td>file</td>
<td>Save</td>
<td>onSave</td>
<td>Save data to text file</td>
</tr>
<tr>
<td></td>
<td>MenuItem_3</td>
<td>3</td>
<td>file</td>
<td>SaveAs</td>
<td>onSaveAs</td>
<td>Save data to text file</td>
</tr>
<tr>
<td></td>
<td>MenuItem_4</td>
<td>4</td>
<td>file</td>
<td>Open</td>
<td>onLoad</td>
<td>Load data from text file</td>
</tr>
<tr>
<td></td>
<td>MenuItem_5</td>
<td>5</td>
<td>file</td>
<td>---------</td>
<td>sep</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>MenuItem_6</td>
<td>6</td>
<td>file</td>
<td>Quit</td>
<td>onQuit</td>
<td>Leave the application</td>
</tr>
</tbody>
</table>

Ready
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 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

<table>
<thead>
<tr>
<th>SolverClass_1</th>
<th>Modified</th>
<th>Class Name</th>
<th>Description Solver_class</th>
<th>Dialog</th>
<th>Log File</th>
<th>Resolution</th>
<th>Derived from Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>solver1</td>
<td>Solver for the Ship problem</td>
<td>CDialgoTSScheduling</td>
<td>solver</td>
<td>10</td>
<td>(null)</td>
</tr>
</tbody>
</table>

| Modified | Application_class | Attribute | Description Variable | Domain Min | Domain Max | |
|----------|-------------------|-----------|----------------------|------------|------------|
| Variable_1 | cumul_task        | start_var | The domain variable for the start of the task | 0          | 100        |
| Variable_2 | cumul_task        | end_var   | The domain variable for the end of the task | 0          | 100        |
| Variable_3 | cumul_task        | duration_var | The domain variable for the duration of the task | 1          | 100        |
| Variable_4 | cumul_task        | resource_used_var | The domain variable for the resource use of the task | 1          | 100        |
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

## Solver Class Table

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class Name</th>
<th>Description</th>
<th>Solver_class</th>
<th>Dialog</th>
<th>Log File</th>
<th>Resolution</th>
<th>Derived from Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverClass_1</td>
<td>2</td>
<td>solver1</td>
<td></td>
<td>CDiallogTSScheduling</td>
<td>solver</td>
<td>10</td>
<td>(null)</td>
</tr>
</tbody>
</table>

## Constraint Table

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Nr</th>
<th>Code</th>
<th>Description Constr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint_1</td>
<td>cumul_task</td>
<td>1</td>
<td>post(end_var == start_var + duration_var)</td>
<td>The precedence constraint</td>
</tr>
<tr>
<td>Constraint_2</td>
<td>cumul_task</td>
<td>2</td>
<td>post(duration_var == solver.convertDurationToDomain(duration))</td>
<td>The length</td>
</tr>
<tr>
<td>Constraint_3</td>
<td>cumul_task</td>
<td>3</td>
<td>post(resource_used_var == resource_used)</td>
<td>The resource usage</td>
</tr>
<tr>
<td>Constraint_4</td>
<td>precedence</td>
<td>4</td>
<td>post(task2.start_var == task1.end_var)</td>
<td>The precedence constraint</td>
</tr>
<tr>
<td>Constraint_5</td>
<td>cumul_machine</td>
<td>5</td>
<td>(limit,0,resources_available); cumul_post(entries,limit,solver.cost)</td>
<td>The cumulative</td>
</tr>
</tbody>
</table>

Ready
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)
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

```cpp
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

```cpp
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, getAvailableResources());
    if (!entries.isEmpty()) {
        ChipCumulative cumul(entries, limit);
        cumul.setEnd(solver->getCost());
        if (ChipPost(cumul) != ChipSucceed) {
            solver->solverError("Constraint failed");
        }
    }
}
```
Generated Code: Precedence

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

- **Application_class**: cumul_task
- **Description Enumeration**: The labeling routine
- **Type**: Value Choice
- **Optimization**: MinMax
- **Partial Search**: Complete
- **Solver_class**: Solver for the Ship

<table>
<thead>
<tr>
<th>Modified</th>
<th>Nr</th>
<th>Attribute</th>
<th>Description Enumeration_item</th>
<th>Method</th>
<th>First() Method</th>
<th>Last() Method</th>
<th>Next() Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration_item_1</td>
<td>1</td>
<td>start_var</td>
<td>Assign start first</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td>Enumeration_item_2</td>
<td>2</td>
<td>duration_var</td>
<td>Then try minimal duration</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td>Enumeration_item_3</td>
<td>3</td>
<td>resource_used_var</td>
<td>At the end fix the resource consumption</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
</tbody>
</table>
Basic Enumeration Routine (Generated code)

```c
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;
}
```

2/27/2008
MinMax Optimization (Generated Code)

```c
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;
    }
}

2/27/2008 Constraint Methodology
Example 2

Chemical Process Scheduling
Problem Description

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

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

- **Overall cumulative constraint limiting nr of parallel tasks**
- **Hard due-date for each task**
- **Cost is sum of earliness**
Problem Description (2)

- **Original description**
  
  [PG95] J.M. Pinto, I.E. Grossmann

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

- **First CP model from V. Liatsos (IC-Parc)**
  - described in
    
    [ELS99] H. El Sakkout
    Algorithm Hybridisation in Constraint Logic Programming
    Tutorial, PACLPI 99, London, April 1999

- **This model used here**
# Object Definition

## Application Class Definition

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class name</th>
<th>Parent</th>
<th>Db_table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DeletedItem</td>
<td>(null)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Machine</td>
<td>(null)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MachineChoice</td>
<td>machine</td>
<td>che_machine_choice</td>
</tr>
<tr>
<td></td>
<td>MachineChoiceTask</td>
<td>task</td>
<td>che_machine_choice_task</td>
</tr>
<tr>
<td></td>
<td>MachineChosen</td>
<td>machine</td>
<td>che_machine_chosen</td>
</tr>
<tr>
<td></td>
<td>Speed</td>
<td>(null)</td>
<td>che_speed</td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>(null)</td>
<td></td>
</tr>
</tbody>
</table>

## Attribute Definition

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Description Attribute</th>
<th>Type</th>
<th>Width</th>
<th>Editable</th>
<th>Default_value</th>
<th>Not_null</th>
<th>Generate GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>machine_rhine_task</td>
<td>machine_rhine</td>
<td>class</td>
<td>9</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>machine_choice_task</td>
<td>resource_var</td>
<td>dvar</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>machine_choice_task</td>
<td>cost_var</td>
<td>dvar</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>machine_choice_task</td>
<td>release</td>
<td>time</td>
<td>4</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>machine_choice_task</td>
<td>cost</td>
<td>integer</td>
<td>9</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
GUI Design

- list
- gantt
- chart
# GUI Definition

<table>
<thead>
<tr>
<th>Modified</th>
<th>Label</th>
<th>Callback</th>
<th>Application_class Top</th>
<th>Menu</th>
<th>Nr</th>
<th>Display_type</th>
<th>Known Callback</th>
<th>Accr</th>
</tr>
</thead>
<tbody>
<tr>
<td>View_1</td>
<td>MachineChoice</td>
<td>OnViewMachineChoice</td>
<td>machine_choice</td>
<td>view</td>
<td>8</td>
<td>List x2</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View_2</td>
<td>MachineSpeed</td>
<td>OnViewMachineSpeed</td>
<td>machine_choice_task</td>
<td>view</td>
<td>9</td>
<td>List x2</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>View_3</td>
<td>MachineGantt</td>
<td>OnViewMachineGantt</td>
<td>machine_choice_task</td>
<td></td>
<td>10</td>
<td>Gantt</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Display Elements

<table>
<thead>
<tr>
<th>Modified</th>
<th>Description</th>
<th>Display_element</th>
<th>Display_type</th>
<th>Resource Class</th>
<th>Resource Label Attribute</th>
<th>Resource Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>MachineGantt</td>
<td>Ganitt</td>
<td>machine_chosen</td>
<td>description</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

#### Menu Items

<table>
<thead>
<tr>
<th>Modified</th>
<th>Description Menu</th>
<th>Label</th>
<th>Nr</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>main</td>
<td>Menubar</td>
<td>1</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>file</td>
<td>&amp;File</td>
<td>2</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>view</td>
<td>&amp;Views</td>
<td>3</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>solve</td>
<td>&amp;Solve</td>
<td>4</td>
<td>main</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified</th>
<th>Menu Item</th>
<th>Nr</th>
<th>Description Menu Item</th>
<th>Callback</th>
<th>Known Callback</th>
<th>Explanation</th>
<th>Accelerator</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>solve</td>
<td>7</td>
<td>Schedule</td>
<td>onSolver</td>
<td></td>
<td>Yes</td>
<td>CTRL + R</td>
<td>Schedule</td>
</tr>
</tbody>
</table>

Ready
### Variable Definition

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

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Attribute</th>
<th>Description Variable</th>
<th>Domain Min</th>
<th>Domain Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>machine_chinese_task</td>
<td>start_var</td>
<td>The domain variable for the start of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>end_var</td>
<td>The domain variable for the end of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>duration_var</td>
<td>The domain variable for the duration of the task</td>
<td>1</td>
<td>20000</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>resource_var</td>
<td>The domain variable for the resource use of the task</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>cost_var</td>
<td>The domain variable for the resource use of the task</td>
<td>0</td>
<td>20000</td>
</tr>
</tbody>
</table>
## Constraint Definition

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Solver_class</th>
<th>Dialog</th>
<th>Log File</th>
<th>Resolution</th>
<th>Derived from Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverClass_1</td>
<td>Solver for the Chemi problem</td>
<td>solver1</td>
<td>CDialgTSScheduling</td>
<td>solver</td>
<td>10</td>
<td>(null)</td>
</tr>
</tbody>
</table>

### Code Examples

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Application class</th>
<th>Nr</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint_1</td>
<td>machine_choice_task</td>
<td>1</td>
<td>post(Ready);max(task1.start_var,task1.duration_var,1);vdiff_post(ready1)</td>
</tr>
<tr>
<td>Constraint_2</td>
<td>machine_choice_task</td>
<td>2</td>
<td>post(start_var == solver1.convertDurationToDomain(release) + cost_var)</td>
</tr>
<tr>
<td>Constraint_3</td>
<td>machine_choice_task</td>
<td>3</td>
<td>post(element(resource_var,table,duration_var))</td>
</tr>
<tr>
<td>Constraint_4</td>
<td>machine_choice_task</td>
<td>4</td>
<td>e_task,entries.add(task.start_var,task.resource_var,task.duration_var,1));diff_post(ready1)</td>
</tr>
<tr>
<td>Constraint_5</td>
<td>machine_choice_task</td>
<td>5</td>
<td>(end1.add(task1.start_var,task1.duration_var,1));dvar(limit,0,parallel);cumul_post(ready1)</td>
</tr>
<tr>
<td>Constraint_6</td>
<td>machine_choice_task</td>
<td>6</td>
<td>(ear(sum);forall(task2,machine_choice_task,sum += task2.cost_var);post(solver.cost == sum))</td>
</tr>
</tbody>
</table>
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

### Constraint Methodology

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application class</th>
<th>Nr</th>
<th>Description Enumeration</th>
<th>Type</th>
<th>Optimization</th>
<th>Partial Search</th>
<th>Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>machine_choice_task</td>
<td>The labeling</td>
<td>Variable Choice</td>
<td>Complete</td>
<td>Solver for the</td>
</tr>
<tr>
<td>Enumeration_1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified</th>
<th>Nr</th>
<th>Attribute</th>
<th>Description Enumeration_item</th>
<th>Method</th>
<th>First() Method</th>
<th>Last() Method</th>
<th>Next() Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>start_var</td>
<td>Assign start first</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>duration_var</td>
<td>Then try minimal duration</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>resource_var</td>
<td>At the end fix the resource</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>cost_var</td>
<td>At the end fix the cost</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
</tbody>
</table>
# Finished Application

## Table

<table>
<thead>
<tr>
<th>Task</th>
<th>Modified</th>
<th>DB Key</th>
<th>Duration</th>
<th>End Date</th>
<th>Machine</th>
<th>Start Date</th>
<th>Cost</th>
<th>Machine Choice</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Choice Task_1</td>
<td>1</td>
<td>1</td>
<td>19h:00</td>
<td>20/07/00 11:00:00</td>
<td>Machine 4</td>
<td>20/07/00 04:00:00</td>
<td>0</td>
<td>Virtual</td>
<td>2500:00</td>
</tr>
<tr>
<td>Machine Choice Task_2</td>
<td>1</td>
<td>2</td>
<td>13:30</td>
<td>12/04/00 11:30:00</td>
<td>Machine 4</td>
<td>07/04/00 00:00:00</td>
<td>0</td>
<td>Virtual</td>
<td>0:00</td>
</tr>
<tr>
<td>Machine Choice Task_3</td>
<td>1</td>
<td>3</td>
<td>13:20</td>
<td>20/06/00 05:20:00</td>
<td>Machine 4</td>
<td>01/06/00 13:20:00</td>
<td>0</td>
<td>Virtual</td>
<td>1333:20</td>
</tr>
<tr>
<td>Machine Choice Task_4</td>
<td>1</td>
<td>4</td>
<td>260:40</td>
<td>03/06/00 01:50:00</td>
<td>Machine 3</td>
<td>23/05/00 23:10:00</td>
<td>1763</td>
<td>Virtual</td>
<td>833:20</td>
</tr>
<tr>
<td>Machine Choice Task_5</td>
<td>1</td>
<td>5</td>
<td>122:40</td>
<td>04/07/00 16:30:00</td>
<td>Machine 3</td>
<td>20/06/00 13:50:00</td>
<td>2035</td>
<td>Virtual</td>
<td>1666:40</td>
</tr>
<tr>
<td>Machine Choice Task_6</td>
<td>1</td>
<td>6</td>
<td>877:10</td>
<td>13/05/00 13:10:00</td>
<td>Machine 1</td>
<td>07/04/00 00:00:00</td>
<td>0</td>
<td>Virtual</td>
<td>0:00</td>
</tr>
<tr>
<td>Machine Choice Task_7</td>
<td>1</td>
<td>7</td>
<td>505:50</td>
<td>28/06/00 13:35:00</td>
<td>Machine 3</td>
<td>08/06/00 12:50:00</td>
<td>0</td>
<td>Virtual</td>
<td>1566:00</td>
</tr>
<tr>
<td>Machine Choice Task_8</td>
<td>1</td>
<td>8</td>
<td>250:00</td>
<td>23/05/00 23:10:00</td>
<td>Machine 3</td>
<td>13/05/00 13:10:00</td>
<td>1263</td>
<td>Virtual</td>
<td>666:40</td>
</tr>
<tr>
<td>Machine Choice Task_9</td>
<td>1</td>
<td>9</td>
<td>311:30</td>
<td>25/04/00 11:30:00</td>
<td>Machine 3</td>
<td>12/04/00 11:30:00</td>
<td>789</td>
<td>Virtual</td>
<td>0:00</td>
</tr>
</tbody>
</table>

## Diagram

- April 2000
- May 2000
- June 2000
- July 2000

- Machine 1
- Machine 2
- Machine 3
- Machine 4

- Task execution timeline with specific dates and durations.
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
Object Model

DeletedItem
- dbKey : int
- className : atom
- classKey : int

Warning
- dbKey : int
- attribute : atom
- explanation : atom
- reference : atom

IntermediateSetup
- dbKey : int
- value : int

Task
- dbKey : int
- start : date
- end : date
- duration : time
- startVar : dvar
- endVar : dvar
- durationVar : dvar

Machine
- dbKey : int
- description : atom
- seqNr : int
- hr : int

MachineChoice
- nrMachines : int
- parallel : int

DisjMachine

MachineChosen

DisjTask

MachineChoiceTask
- resourceVar : dvar
- qty : int

resource

IntermediateProduct
- dbKey : int
- batchSize : int
- color : color
- description : atom
- initialStock : int
- speed : int

to

from

IntermediateSetup

IntermediateProduct
- dbKey : int
- color : color
- description : atom
- initialStock : int
- speed : int

FinishedProduct
- dbKey : int
- value : int

ProducerConsumer

Producer

Consumer

Speed
- dbKey : int
- value : time

Constraint Methodology

2/27/2008
Object Definition

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class_name</th>
<th>Parent</th>
<th>Db_table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationClass_1</td>
<td>DeletedItem</td>
<td>(null)</td>
<td>-</td>
</tr>
<tr>
<td>ApplicationClass_2</td>
<td>DisjMachine</td>
<td>machine</td>
<td>bs_disj_machine</td>
</tr>
<tr>
<td>ApplicationClass_4</td>
<td>DisjTask</td>
<td>task</td>
<td>bs_disj_task</td>
</tr>
<tr>
<td>ApplicationClass_6</td>
<td>FinishedProduct</td>
<td>(null)</td>
<td>bs_finished_product</td>
</tr>
<tr>
<td>ApplicationClass_7</td>
<td>IntermediateProduct</td>
<td>(null)</td>
<td>bs_intermediate_product</td>
</tr>
<tr>
<td>ApplicationClass_8</td>
<td>IntermediateSetup</td>
<td>(null)</td>
<td>bs_intermediate_setup</td>
</tr>
<tr>
<td>ApplicationClass_3</td>
<td>Machine</td>
<td>(null)</td>
<td>-</td>
</tr>
<tr>
<td>ApplicationClass_9</td>
<td>MachineChoice</td>
<td>machine</td>
<td>bs_machine_choice</td>
</tr>
</tbody>
</table>

This class describes a super-class from which all machines inherit.

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Description</th>
<th>Attribute</th>
<th>Type</th>
<th>Width</th>
<th>Editable</th>
<th>Default_value</th>
<th>Not_null</th>
<th>Generate GUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute_14</td>
<td>DisjTask</td>
<td>qty</td>
<td>integer</td>
<td>9</td>
<td>Yes</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Attribute_15</td>
<td>DisjTask</td>
<td>intermediate_product</td>
<td>class</td>
<td>9</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Ready
GUI Description

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

<table>
<thead>
<tr>
<th>Description</th>
<th>Display Type</th>
<th>Resource Class</th>
<th>Resource Label Attribute</th>
<th>Resource Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayElement_1</td>
<td>MachineGantt</td>
<td>Gantt</td>
<td>machine</td>
<td>description</td>
</tr>
<tr>
<td>DisplayElement_2</td>
<td>IntermediateStock</td>
<td>Chart</td>
<td>producer_consumer</td>
<td>description</td>
</tr>
</tbody>
</table>
### Menu Description

**Table 1: Menu Description**

<table>
<thead>
<tr>
<th>Modified</th>
<th>Menu</th>
<th>Nr</th>
<th>Description Menu</th>
<th>Label</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>main</td>
<td>1</td>
<td>MenuBar</td>
<td>main</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>file</td>
<td>2</td>
<td>&amp;File</td>
<td>main</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>view</td>
<td>3</td>
<td>&amp;Views</td>
<td>main</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>baseview</td>
<td>4</td>
<td>&amp;Views</td>
<td>main</td>
<td>main</td>
</tr>
<tr>
<td>2</td>
<td>solve</td>
<td>5</td>
<td>&amp;Solve</td>
<td>main</td>
<td>main</td>
</tr>
</tbody>
</table>

**Table 2: View Description**

<table>
<thead>
<tr>
<th>Modified</th>
<th>Menu</th>
<th>Nr</th>
<th>Description Menu_item</th>
<th>Callback</th>
<th>Known Callback</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>view</td>
<td>8</td>
<td>MachineChnise</td>
<td>OnViewMachineChnise</td>
<td>Yes</td>
<td>menu belonging to view</td>
</tr>
<tr>
<td>2</td>
<td>view</td>
<td>9</td>
<td>Speed</td>
<td>OnViewSpeed</td>
<td>Yes</td>
<td>menu belonging to view</td>
</tr>
<tr>
<td>2</td>
<td>view</td>
<td>10</td>
<td>IntermediateProduct</td>
<td>OnViewIntermediateProduct</td>
<td>Yes</td>
<td>menu belonging to view</td>
</tr>
<tr>
<td>2</td>
<td>view</td>
<td>11</td>
<td>Setup</td>
<td>OnViewSetup</td>
<td>Yes</td>
<td>menu belonging to view</td>
</tr>
</tbody>
</table>

Ready
### Variable Description

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class Name</th>
<th>Description Solver_class</th>
<th>Dialog</th>
<th>Log File</th>
<th>Resolution</th>
<th>Derived from Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>SolverClass_1</td>
<td>2</td>
<td>solver1</td>
<td>Solver for the BASF problem</td>
<td>CDialgTSScheduling</td>
<td>solver</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Attribute</th>
<th>Description Variable</th>
<th>Domain Min</th>
<th>Domain Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable_1</td>
<td>machine_chinate_task</td>
<td>start_var</td>
<td>The domain variable for the start of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>Variable_2</td>
<td>machine_choice_task</td>
<td>end_var</td>
<td>The domain variable for the end of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>Variable_3</td>
<td>machine_choice_task</td>
<td>duration_var</td>
<td>The domain variable for the duration of the task</td>
<td>1</td>
<td>20000</td>
</tr>
<tr>
<td>Variable_4</td>
<td>machine_choice_task</td>
<td>resource_var</td>
<td>The domain variable for the resource use of the task</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Variable_1</td>
<td>disi_task</td>
<td>start_var</td>
<td>The domain variable for the start of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>Variable_2</td>
<td>disi_task</td>
<td>end_var</td>
<td>The domain variable for the end of the task</td>
<td>0</td>
<td>20000</td>
</tr>
<tr>
<td>Variable_3</td>
<td>disi_task</td>
<td>duration_var</td>
<td>The domain variable for the duration of the task</td>
<td>1</td>
<td>20000</td>
</tr>
</tbody>
</table>
## Constraint Description

### Solver Class

<table>
<thead>
<tr>
<th>Modified</th>
<th>Class Name</th>
<th>Description</th>
<th>Solver Class</th>
<th>Dialog</th>
<th>Log File</th>
<th>Resolution</th>
<th>Derived from Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Solver1</td>
<td>Solver for the BASF problem</td>
<td>CDiallogTSScheduling</td>
<td>solver</td>
<td></td>
<td>10</td>
<td>(null)</td>
</tr>
</tbody>
</table>

### Application Classes

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application Class</th>
<th>Nr</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>disj_task</td>
<td>1</td>
<td>post((\text{end_var} == \text{start_var} + \text{duration_var}))</td>
</tr>
<tr>
<td>2</td>
<td>disj_task</td>
<td>2</td>
<td>post((\text{duration_var} == \text{solver_convertDurationToDomain}(\text{duration})))</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>1</td>
<td>post((\text{end_var} == \text{start_var} + \text{duration_var}))</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>3</td>
<td>\text{ver_convertDurationToDomain}(\text{speed_value}))\post((\text{element}(\text{resource_var}_\text{table}_\text{duration_var})))</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>4</td>
<td>\text{add}(\text{task}_\text{start_var}_\text{task}_\text{resource_var}_\text{task}_\text{duration_var},1)\text{_diffn}_\text{element}(\text{entries})</td>
</tr>
<tr>
<td>2</td>
<td>machine_choice_task</td>
<td>5</td>
<td>\text{add}(\text{task}_1_\text{start_var}_\text{task}_1_\text{duration_var},1)\text{_dvar}(\text{limit},0_\text{nr_machines})\text{_cumul}_\text{post}(\text{entries1}_\text{limit})</td>
</tr>
<tr>
<td>2</td>
<td>disj_machine</td>
<td>6</td>
<td>\text{add}(\text{task}_1_\text{start_var}_\text{task}_1_\text{duration_var},1)\text{_dvar}(\text{limit},0,1)\text{_cumul}_\text{post}(\text{entries1}_\text{limit})</td>
</tr>
<tr>
<td>2</td>
<td>producer_consumer</td>
<td>5</td>
<td>producer_consumer((\text{disj_task}_\text{machine_choice}_\text{task}))</td>
</tr>
</tbody>
</table>
Constraint Model

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

\[
\text{post}(\text{end\_var} == \text{start\_var} + \text{duration\_var})
\]

\[
\text{post}(\text{duration\_var} == \text{solver.convertDurationToDomain(duration)})
\]
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

```plaintext
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

### Table 1: Constraint Methodology

<table>
<thead>
<tr>
<th>Modified</th>
<th>Application_class</th>
<th>Nr</th>
<th>Description Enumeration</th>
<th>Type</th>
<th>Optimization</th>
<th>Partial Search</th>
<th>Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration_1</td>
<td>disi_task</td>
<td>1</td>
<td>The labeling routine</td>
<td>Snake</td>
<td>None</td>
<td>Complete</td>
<td>Solver for the...</td>
</tr>
<tr>
<td>Enumeration_2</td>
<td>machine_choice_task</td>
<td>2</td>
<td>The labeling routine</td>
<td>Value Choice</td>
<td>None</td>
<td>Complete</td>
<td>Solver for the...</td>
</tr>
</tbody>
</table>

### Table 2: Enumeration Item Details

<table>
<thead>
<tr>
<th>Modified</th>
<th>Nr</th>
<th>Attribute</th>
<th>Description Enumeration_item</th>
<th>Method</th>
<th>First() Method</th>
<th>Last() Method</th>
<th>Next() Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerationitem_1</td>
<td>1</td>
<td>start_var</td>
<td>Assign start first</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td>Enumerationitem_2</td>
<td>2</td>
<td>duration_var</td>
<td>Then try minimal duration</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
<tr>
<td>Enumerationitem_3</td>
<td>2</td>
<td>resource_var</td>
<td>At the end fix the resource</td>
<td>Min</td>
<td>unused</td>
<td>unused</td>
<td>unused</td>
</tr>
</tbody>
</table>
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