A Case of Business Model Conversion Using Planning and Scheduling Application Having Intermediate Cutting Stock*

Masayoshi TAKADA**

This paper describes the experience of developing a production planning and scheduling system, which makes more than 1700 kinds of end products from more than 300 kinds of large plates stocked in the intermediate warehouse. This system was mostly achieved with Constraint Logic Programming. As our business environments having changed, we were in urgent need of converting our production method to the pull-type from the push-type. With the push-type production method, we have to keep our finish products in stock for three to four months and refill the stock in 30 days. The features of our system: it solved the complex network problem of 38 patterns found in the whole production process from cutting to shipment in generic way. At the former process, it also solved an optimization of the setups problem in the two-phase batch-type production.

Key Words: Constraint Logic Programming, Business Process Optimization, Cutting Stock Problem, Scheduling, Planning, APS, SCM

1. Introduction

PS plate (Pre-sensitized Plate) is a material used for the offset lithography. Its support body is a thin aluminum board and it has chemical treatment on the surface so that it has enough strength and good water retentiveness. The photosensitizer is spread on this surface. That is how the products made.

It is a material for printed matter such as newspaper, magazine, advertisement, etc. and the end users exceed 3,000 nationwide. Also there is much demand for Just In Time delivery, so we came to change our production system. A big opportunity came when a big user withdrew from the PS Plates Business.

The followings are the main features of the new production system. (1) Support for the conversion to the pull-type from the push-type, (2) large plates are used as intermediate products for strategic production management, (3) The judging criteria are set for shortening our total lead time and makespan from cutting to wrapping and shipment that contains complex routes (called 'Process Path'). As for the former process, a criterion of setup cost between treatment kinds is set.

2. The Production System

The production system of the PS plate is described as follows. (1) Purchase the aluminum coil as a raw material. (2) Start the 24-hour continuous process of grainling and photosensitizer spreading on one line. (3) Keep PS plates in the semi-manufactured warehouse in the form of large plates. (4) Cut them into rectangular products according to the shipment plan, (5) Wrap, pack and ship them to a local warehouse by truck.

The features and constraints of each process are shown as follows.

2.1 The former process (grain and spreading)

Each coil is different in width and thickness. We have to set up the machines for each coil; setting on the machine, cleaning etc. It depends on width of a replacing coil, in detail it takes a few minutes for replacing to a narrower coil or conversely more than 15
minutes for replacing to a wider coil from current state.

To create a grain on the surface, we apply chemical and electric treatment on it. A different grain requires different type of treatment, so we change how to operate machines every time we switch grains. This will take 30–90 minutes. The photosensitizer spreading begins after the aging time for chemical and electro-treatment. The coil with photosensitizer on it is cut into semi-manufactured products (large plate) and distributed to the nondefective and defective products automatically at the end of the spreading process. A large plate is piled up on the palette, and kept in the middle warehouse using the forklifts.

2.2 Semi-manufactured (large plate and selection)

After separating defective large plates from good ones, we check the defective ones again with our own eyes. Only goods plates approved by our quality assurance people are kept in the large plate automated warehouse. But there is a limit to the number of plates we can keep. We can keep plates there for about half a year for the longest.

2.3 Post-processing (cutting, wrapping, and shipment)

First, select large plates and cut them into a required size and number on the order specifications. Then, transport them up to the next process using the palette. After that, wrap a specified number of boards into one packet and put them on the palette. Finally deliver them to each destination by truck.

In the post-processing, there are seven paper cutters, four shape correction stands, two combination paper pulling out, one automatic packer, two manual packing machines, a Boxed Types packing machines, two length bulk packing, and flat bulk packing and newspaper packing. After passing through these machines, products get to the final stage of the whole process, shipping. These routes are called Process Path.

In the next paragraph, each machine of the post-processing and the handling of process passing are explained in full.

2.4 Process path and machine of post-processing

Each machine is called Disjunctive Machine, for the machine that does not necessarily pass excluding the paper cutter and the shipment. The paper cutter is called Machine Choice Machine because you can choose a suitable one from plural machines. Additionally, Box Type, Pallet Type, Selection Line, and Manpower were introduced. The Table 1 shows these machine types.

These machines are abstractions of the physical machines found in the factory, and do not reflect all the details of the real machines. Their purpose is to capture those constraints that have a real impact on the quality of the schedule. The scheme of the typical machine type is described as follows*1.

2.4.1 Machine Choice The MachineChoice machine exists together with the MachineChosen objects represent the cutting section in the factory. All MachineChosen instances refer to this MachineChoice object. The MachineChoice object represents all of the cutting machinery, while each MachineChosen object represents one cutting machine. Each pallet of intermediate boards that is processed by a cutting operation creates one task on the MachineChoice machine. Each MachineChosen object has an attribute nr, these must be all different and must be numbered from 1 to K, where K is the number of MachineChosen objects.

In the ProcessStep objects, we must always refer to the MachineChoice object, never to a MachineChosen instance. On the other hand, all cutting tasks are assigned to MachineChosen resources, which indicates on which physical cutting machine a task is scheduled.

If one of the ProcessSteps of an order uses the MachineChoice resource, then we have to generate one task for each pallet that is allocated to the order.

The constraint of the MachineChoice object is expressed by a diff constraint. In addition to the start and end domain variables, each task on the cutting machine also has a machineVar domain variable. The domain of that variable ranges from 1 to K. Furthermore, each task has a machine preference domain variable machinePreferenceVar, whose domain is 1 to 20. This preference variable is used to express constraints on which particular cutting machine should be used for some task (see below).

The diff constraint takes all tasks that perform some cutting operation, i.e., all tasks whose processStep attribute uses the MachineChoice object and

<table>
<thead>
<tr>
<th>Table 1 Developed machine types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Choice</td>
</tr>
<tr>
<td>DisjMachine</td>
</tr>
<tr>
<td>CumalMachine</td>
</tr>
<tr>
<td>Manpower</td>
</tr>
<tr>
<td>PalletTypeMachine</td>
</tr>
<tr>
<td>BoxTypeMachine</td>
</tr>
<tr>
<td>SelectionLine</td>
</tr>
</tbody>
</table>

*1 These machines have been created using a few global and generic constraint framework of constraint logic programming to handle specific constraints among production processes. That is why we have different machine type.
states the condition:

dfnf ([Start_working_var1,
    Machine_var1, Working_duration1, 1],
    [Start_working_var2, Machine_var2, Working_duration2, 1,...]).

This declaratively states the condition for any two tasks \( i \) and \( j \):

\[
\forall (start\_work\_ing\_var \geq start\_work\_ing\_var,
    \begin{align*}
    + & \text{ working\_du\_ration\_var} \\
    \vdash & \text{ start\_work\_ing\_var} \\
    + & \text{ working\_du\_ration\_var} \\
    \vdash & \text{ machine\_va} r_j \equiv \text{machine\_va} r_i
    \end{align*}
\]  

(1)

The constraint is expressed in the working time co-ordinate system. Intuitively, this constraint makes sure that two different tasks are not overlapping, i.e. each cutting machine performs one task at a time. It also ensures that tasks are non-preemptive, i.e. they cannot interrupt each other. Considering all cutting machines together, there can be as many cutting tasks running in parallel, as there are cutting machines. The duration of the tasks on the cutting machines is given by the value provided in the speed attribute of the \text{CutType} object used by each task. This speed is given in minutes per case. Graphically, the \text{MachineChoice} machine can be represented by Fig. 1.

Not all products can be cut on all cutting machines with the same preference. For each task on a cutting machine, we can define which cutting machine is preferred and/or possible. The checking of these conditions is done by a hard-coded procedure written in C++, which returns a list of preference values for each machine. If a machine is not possible (indicated by the ‘\( \vdash \)’ in the Table 2) a value of 21 will be returned in the list, which makes the corresponding machine infeasible*2. The returned value is the sum of all preference values from the Table 2, i.e. each condition is tested and if applicable, the value for each machine is added. A small preference value states a high preference, i.e. this machine will be tried first. For each task on the cutting machine, an element constraint links the \text{machineVar} and the \text{MachinePreferenceVar} of the task.

element \( (\text{Machine}_\text{var}, [...\text{Preference values...}],
    \text{MachinePreferenceVar}) \)

The above conditions require that the \text{MachineChosen} instances are numbered 1, 2, 3, 4, 5, 6, 7 for machines A, B, C, D, E, F, G.

2.4.2 Disjunctive Machine

The disjunctive machine is a much simpler machine type than the machine choice. Graphically, the constraint can be expressed as:

The machine can run one task at a time. Tasks cannot interrupt each other. The duration of each task is given, which is calculated on the number of pieces/cases/pallets used by the task, and the speed and the \text{speedUnit} that is given in the \text{processStep} of the task. If one of the \text{ProcessSteps} of an order uses a \text{DisjMachine} resource, then we have to generate one task for each pallet that is allocated to the order. The constraint is expressed with a cumulative constraint using a resource limit of 1. For all tasks using the machine the constraint is expressed as

\[
\forall i \in \text{end_array}, \text{Working\_horizon}, \text{cumulative}(\begin{align*}
    \text{Start\_working\_var1},
    \text{Start\_working\_var2}, \ldots,
    \text{Working\_duration\_var1},
    \text{Working\_duration\_var2}, \ldots, [1, 1, \ldots, \text{unused}, \text{unused}, 1, \text{End}],
\end{align*})
\]

which expresses declaratively that for any two different tasks \( i \) and \( j \):

\[
\forall (\begin{align*}
    \begin{align*}
    \vdash & \text{ working\_du\_ration\_var} \\
    \vdash & \text{ start\_work\_ing\_var} & \vdash & \text{ start\_work\_ing\_var}
    \end{align*}
    \end{align*}
\]

Fig. 2 Disjunctive machine constraint

*2 “21” is a value in the case that a machine is not possible, because a summation of the preferences for each machine always under the value if the machine available.
In addition, the equation

\[ \text{End} = \max \left( \text{Start\_working\_var}, \text{working\_du\_ration\_var} \right) \tag{3} \]

holds.

### 2.4.3 Cumulative Machine

The CumulMachine type is slightly more complex. This type of machine can run more than one task, if the total amount of work is kept below a given resource limit at all times. Tasks can use one or more resource units, but always the same amount from start to finish. Depending on the number of resources used, the duration of the task will vary. The constraint is that the product of duration and resources used is fixed and corresponds to the overall amount of work to be performed. Duration and resources used for each task are integer values. Tasks on this type of machine require additional attributes. The domain variable `resources\_used\_var` denotes the number of resources used, the attribute surface is an integer value which holds the overall amount of work required for the task. The following constraint holds:

\[ \text{Surface} = \text{Resources\_used\_var} * \text{Working\_duration\_var} \tag{4} \]

The overall number of resources available is given as an attribute of the CumulMachine itself. If one of the ProcessSteps of an order uses a CumulMachine resource, then we have to generate one task for each pallet that is allocated to the order. The constraint is expressed with the cumulative constraint:

\[ \text{Limit}: 0..\text{Overall\_resources} \]
\[ \text{End}: 0..\text{Working\_horizon} \]
\[ \text{cumulative}([\text{Start\_working\_var1, Start\_working\_var2, ...}], [\text{Working\_duration\_var1, Working\_duration\_var2, ...}], [\text{Resources\_used\_var1, Resources\_used\_var2, ...}], \text{unused}, [\text{Surface1, Surface2, ...}], \text{Limit, End}) \]

Declaratively, the constraint expresses

\[ \max \left( \sum_{\text{Start\_working\_var} + \text{working\_du\_ration\_var}} \text{resources\_used\_var} \right) = \text{Limit} \tag{5} \]

In addition, the equation

\[ \text{End} = \max \left( \text{Start\_working\_var}, \text{working\_du\_ration\_var} \right) \tag{6} \]

holds. Graphically, the constraint can be expressed as Fig. 3.

In this figure, the profile means an upper limit constraint of resource use, which should be expressed from outer systems having constraints among the resources. The alternatives mean that we have the possibilities to put the schedule of those tasks over time in the given constraints.

### 2.4.4 Manpower

The manpower machine type is special since these requirements can be expressed for all tasks in the cut/pack schedule in addition to the resource, which is normally allocated to the task. In the KAPSS system, we can handle exactly two different manpower resources, male and female workers. For each task, a different number of workers may be needed. This manpower need applies during the whole duration of the task. The total amount of workers of each type is limited, the limit is given in the `resources\_available` attribute of the object. The manpower resources are also special in that there is no calendar which is applied for them. The only limit enforced is that the total number of workers at each time point is not exceeded. The start and end dates for all tasks are the elapsed time variables, as the manpower resources apply to all tasks on all machines and the elapsed time is the only common co-ordinate system. For each manpower resource, in principle we have to include as many tasks as we have `CutPackTask` and `SelectionTask` objects in the schedule. Obviously, tasks with zero resource requirements do not need to be represented. We can use an estimate of the number of tasks that are needed.

### 2.4.5 Process Path

At the moment, 38 process paths are defined for the different products handled by the cutting/packing machines. For each order, only one process path is applicable. This depends on the size of the board, the finished product and the "requirement-code", an order specific description of special customer demands. This code describes special requirements, which are reflected in a particular process path. The details of the "requirement-code" and its impact on the process path selection are described below.

The machine speed depends on the process path and the machine used. For the different machines, the speed is given in different units. The speed and the speed unit is given in the `ProcessStep` objects. Note that for cutting (machine 0), the speed is calculated from the cut type.

The transportation times are always added at the
end of an activity and are stored in the ProcessStep objects. The initial transport time from the warehouse to the cutting is not stored in a ProcessStep, but is a general parameter.

CutPackTasks are activities that are created for each order for all ProcessSteps that are given in the ProcessPath of the order. The number of tasks for one order depends either on the intermediate board pallet allocation or on the number of finished product pallets in the order. Each CutPackTask also contributes to the overall manpower constraints which also include the selection tasks.

The picture summarizes the material flows. Where the numbers (0 to 16) means the process order and machine in the factory described at Table 3. Any process path has process step of machine 0 (cutting) and machine 16 (shipping), which is basic structure of the process paths (A01, .., D1).

2.4.6 Evaluation We have shortened our order refill lead-time to the current 2.6 days from 30 days. We have also reduced the amount of inventory by 45% since the introduction of our new system. The frequency of rescheduling has increased to 2-3 times a day from once a week.

3. Summary

We have developed a scheduling system as general as possible so that it can satisfy the complex pull-type constraint conditions. The performance of the system is presented in Table 5. We have managed to save about 30,000,000 yen\(^*7\) a year on distribution cost because we have succeeded in solving our distribution problems, by using backward scheduling, which establishes a schedule backward from a designated due date shipping time to an appropriate start day. It was impossible to make a schedule backward manually. Improving our distribution was crucial to the success of our project.

Shortly, we could summarize the project as following.

(1) Because of a drastic change in the landscape of the PS plates industry, we were in urgent need of developing a new system that supports the conversion to the pull-type production method from the push-type constraint conditions. The performance of the system is presented in Table 5. We have managed to save about 30,000,000 yen\(^*7\) a year on distribution cost because we have succeeded in solving our distribution problems, by using backward scheduling, which establishes a schedule backward from a designated due date shipping time to an appropriate start day. It was impossible to make a schedule backward manually. Improving our distribution was crucial to the success of our project.

Table 3  Process steps

<table>
<thead>
<tr>
<th>No</th>
<th>Machine</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Machine choice</td>
<td>Cutting</td>
</tr>
<tr>
<td>1</td>
<td>Disjunctive</td>
<td>Drilling</td>
</tr>
<tr>
<td>2</td>
<td>Alt-Machine</td>
<td>Shape correction</td>
</tr>
<tr>
<td>3</td>
<td>Disjunctive</td>
<td>Paper removal</td>
</tr>
<tr>
<td>5</td>
<td>Disjunctive</td>
<td>Newspaper packing</td>
</tr>
<tr>
<td>6</td>
<td>Disjunctive</td>
<td>Auto packer</td>
</tr>
<tr>
<td>7</td>
<td>Disjunctive</td>
<td>Manual packer A</td>
</tr>
<tr>
<td>8</td>
<td>Disjunctive</td>
<td>Manual packer B</td>
</tr>
<tr>
<td>9</td>
<td>Disjunctive</td>
<td>Manual packer A</td>
</tr>
<tr>
<td>10</td>
<td>Alt-Pallet type</td>
<td>Vertical bulk packing</td>
</tr>
<tr>
<td>11</td>
<td>Pallet type</td>
<td>Flat bulk packing</td>
</tr>
<tr>
<td>12</td>
<td>Box type</td>
<td>Box type packing</td>
</tr>
<tr>
<td>16</td>
<td>Pallet type</td>
<td>Shipping</td>
</tr>
</tbody>
</table>

Table 4  Process paths

<table>
<thead>
<tr>
<th></th>
<th>0</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>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A02</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A03</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B04</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A04</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A08</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C01</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A11</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A12</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D34</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5  Evaluation

<table>
<thead>
<tr>
<th>Item</th>
<th>Before</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order fulfillment</td>
<td>30 days</td>
<td>2.6 days(^*4)</td>
</tr>
<tr>
<td>Product stock</td>
<td>1.0</td>
<td>0.55(^*5)</td>
</tr>
<tr>
<td>Re-schedule cycle</td>
<td>Weekly</td>
<td>2-3 times/day(^*6)</td>
</tr>
</tbody>
</table>

\(^*4\) The lead-time of order fulfillment had been defined as a metric that measures how the business changes. When we had the old business style, we fulfilled the orders so that they are satisfied at the end of each month for summarizing production orders to a user's depot, but currently we have been delivering the well-timed orders for our local stock points. The fewer the value is, the more efficient the business will go. That was how the metrics was set.

\(^*5\) At the same time, we had improved the capacity of intermediate stock up to 160%. This means that we prefer an intermediate product to a finish product.

\(^*6\) The impact of improving re-schedule cycle is obviously that the system have a potential of removing the intermediate delivery process at the depot by using backward re-scheduling, where a delivery time of shipping track is used as due dates of production order. Without the system, we could hardly have done it.

\(^*7\) The cost was almost 10,000 yen/pallet a unit days in order to do temporal delivery for central product stock near factory.
type. The new pull-type method allows us to meet any delivery dates requested in individual order while with the push-type we have to carry greater inventories in the warehouse, for we receive the monthly package order from a big user.

(2) To prevent the risk caused by mismanaging the inventory of our products. We decided to strengthen our risk management abilities. We have developed a new system which support the production made based on order forecasts that seems like the production by order. We have also succeeded in shortening out lead-time through a proper inventory management.

(3) With the satisfaction of all these complicated conditions and the successful implementation of our new system, we now believe that we have established a technological foundation for our Supply Chain Management.

Acknowledgments

We wish to express our gratitude to people in COSYTEC who cooperates with the KAPSS project.

Appendix 1: About the Diffn Global Constraint

The diffn constraint can be applied for many types of packing and placement problems in one or more dimensions. Figure 5 shows some examples of its use. The one-dimensional case expresses cutting stock problems (Fig. 5(A)). In two dimensions this constraint also corresponds to cutting problems of various forms (Fig. 5(C)). In three dimensions, typical problems are packing of palettes or containers with rectangular boxes (Fig. 5(D)). The diffn constraint is also applied for scheduling and assignment problems, where it can easily model disjunctive machine assignment for tasks. The x-dimension of the constraint represents time, each value in the y-dimension stands for a possible machine assignment. Each task is represented as a rectangle, which is given by its start time, its resource assignment, its duration and a height 1 (see Fig. 5(B)). The non-overlapping condition states that no tasks can be scheduled at the same time on the same machine. The difference to the use of the cumulative constraint lies in the fact that we do not only control that enough resources are available globally, but that we assign each task to a particular machine. This type of constraint also occurs in time tabling problem and personnel assignment situations.

Appendix 2: The Advantages of CLP

CLP (constraint logic programming) aims at combining the declarative aspects of logic programming and constraint solving in an efficient problem-solving environment. Constraints over different domains can be stated in a uniform framework and are solved with methods originating in various areas, from artificial intelligence (AI) to Operations Research (OR). CLP is now used industrially for applications as diverse as digital circuit design, portfolio management and production scheduling systems and there are several industrial systems available for constraint programming.

Combinatorial problems occur in many different application domains. For example, scheduling and assignment, placement and layout, financial decision making (option trading or portfolio management) or even biology (DNA sequencing). In all these problems we have to choose among many possible alterna-
tives to find solutions respecting a large number of constraints. We may be asked to find an admissible, feasible solution to a problem or to find a good or even optimal solution according to some evaluation criteria.

From a computational point of view, we know that most of these problems are difficult. They belong to the class of NP-hard problems. This means that no efficient and general algorithms are known to solve them.

At the same time, the problems themselves are rapidly changing. For example, in a factory, new machines with new characteristics may be added, which might completely change the production scheduling problem. New products with new requirements will be added. If a scheduling system cannot be adapted to these changes, it will rapidly become useless. Another aspect is that the complexity of the problems is steadily increasing. This may be due to increased size or complexity of the problem, or may be caused by higher standards on quality or cost effectiveness. Many problems which have been handled manually for a long time now exceed the capacity of a human problem solver.

At the same time, humans are typically very good at solving these complex decision making problems. They can have a good understanding of the underlying problem and often know effective heuristics and strategies to solve them. In many situation they also know which constraints can be relaxed or ignored when a complete solution is not possible. For this reason, it is necessary to build systems which cooperate with the user via friendly graphical interfaces and which can incorporate different rules and strategies as part of the problem solving mechanism. Developing such applications with conventional tools has a number of important drawbacks.

• The development time will be quite long. This will increase the cost, making bespoke application development very expensive.

• The programs are hard to maintain. Due to their size and complexity, a change in one place may well lead to a number of other changes which are required in other places.

• The programs are difficult to adopt and to extend. As new requirements arise during the life cycle of the program, changes become more and more difficult.

The use of constraint logic programming should lead to a number of improvements:

• The development time is decreased, as the constraint solving methods are reused.

• A declarative problem statement makes it easier to change and modify programs.

• As constraints can be added incrementally to a system, new functionality added without changing the overall architecture of the system.

• Strategies and heuristics can be easily added or modified. This allows to include problem specific information inside the problem solver.

References