

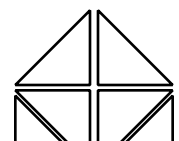
n o t d e f i n e d .

CCSP /

Continuous Caster Sequence Planner

fr om

A. Jablonk a, J. Lemke, U. Pick artz



Content

| | |
|---|-----------|
| 1 SUMMARY | 3 |
| 2 DESCRIPTION OF A STEEL PLANT | 4 |
| 3 WHAT DO WE WANT TO PLAN? | 6 |
| 4 SHORT DESCRIPTION OF CCSP | 9 |
| 5 MAPPING THE STEEL PLANT TO AN OBJECT STRUCTURE | 10 |
| 5.1 PLANT | 11 |
| 5.2 PLANNING OBJECTS | 12 |
| 5.3 CONSTRAINTS | 13 |
| 5.4 ASSIGNMENTS AND STRATEGIES | 14 |
| 6 PORTABILITY PROBLEMS | 17 |

1 Summary

We have developed a software system for planning sequences in a steel plant - this includes rough time scheduling.

The following resources can be taken into account: converters, secondary steelmaking equipment, continuous slab casters.

The design of the steel plant, planning strategies, soft constraints which should be fulfilled as well as hard constraints which must be fulfilled are read from a database. Hence, our system can easily be adapted to different steel plants.

The planning horizon for "detailed (on line) planning" is one day.

Our system CCSP is part of the BRONER system (Broner Group Ltd., Watford, England) which is a production planning system for the whole steel plant.

2 Description of a steel plant

In Fig.1 we have given a schematic view of a traditional steel plant. The following production steps are performed:

- Hot metal is produced in a furnace.
- The converters transform „hot metal + scrap + additions“ to hot steel. The content of one converter is named heat.
- The heat is poured into an empty ladle.
- The ladles are transported on cars and cranes to the secondary metallurgy equipment, e.g. ladle treatment station (LTS) and vacuum degassing (VD).
- The ladles are transported to the continuous slab casters, where hot steel is converted to strands, which are cooled with water until the strands are completely solidified. At the end of the casting process the solidified strand is cut into slabs.
- In the mill coils are made out of slabs.

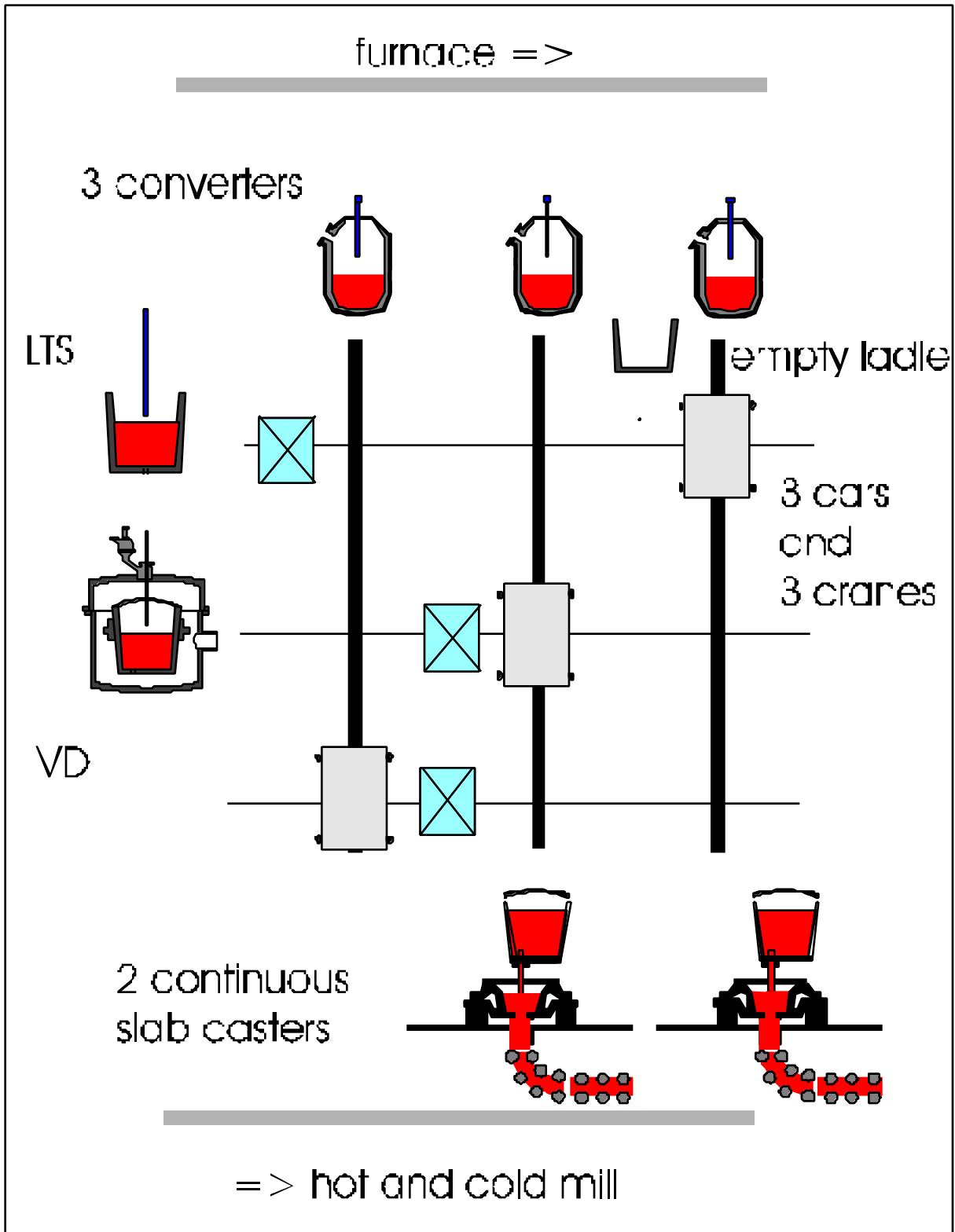


Fig.1: Schematic view of a traditional steel plant.

3 What do we want to plan?

We want to plan the production process from the converters to the continuous slab casters. This planning includes "rough time scheduling", i.e. scheduling without taking into account the cars and cranes.

One input for our system are slab orders. In Fig.2 we have displayed a quite simplified situation where we have 16 different orders. The colors - i.e. violet, blue, and yellow - indicate, that the slabs are made from different steel grades. Now we have to find out how these slabs can be produced, i.e. we have to build up strands and heats. Moreover we have to make sure that resources are available for producing the heats.

The final planning has to fulfill several constraints. We distinguish between four different types:

- **Technical constraints** due to the technical equipment, e.g. a heat has a weight of 220 t, the width of the strand on caster 2 has to be in the domain [2.0,5.0] m.
- **Heuristic constraints** which should be fulfilled to reach a reasonable planning, e.g. slabs with an early due date should be cast first, if possible.
- **Rough scheduling constraints** due to capacity restrictions.
- **Customer constraints** given by the order data, i.e. the width of slab SO4711 has to be in the domain [3.0, 3.2] m, the production grade should be of very high quality.

Moreover we want to minimize certain cost functions.

In the following we discuss the planning results which are displayed in Fig.2.

- All slabs are planned, we get four heats. Two yellow heats, one violet heat, and one blue heat.
- The green slabs are uncommissioned, they are introduced for filling up the heat (a heat consists of a certain amount of hot steel, e.g. 220 t). Moreover the green wedges make sure that the strand width is a continuous function with respect to length. The amount of "green material" should be as small as possible.
- One violet slab is put into the blue heat. Hence the customer who has ordered this violet slab will get a blue one, instead. This is possible because in our example the blue production grade has a better steel quality and it is an allowed substitute. The alternative would have been to introduce a second violet heat. However, it would contain only one ordered slab, and we would have to fill up the heat with uncommissioned slabs. Thus, it is cheaper for the steel plant to deliver this slab with a better quality.
- The heats are ordered in two sequences. Sequence 1 is cast on caster 1 which has two strands and sequence 2 is cast on caster 2, a one strand caster.
- In table 1 we have displayed a heat scheduling plan.

| | Conv 1 | Conv 2 | Conv 3 | LTS | VD | Caster 1 | Caster 2 |
|---------------|--------|--------|--------|-------|-------|----------|----------|
| Heat 1 | 13:00 | | | 14:00 | 14:20 | 15:30 | |
| | 13:45 | | | 14:15 | 15:20 | 16:30 | |
| Heat 2 | 14:14 | | | | 15:20 | 16:30 | |
| | 15:00 | | | | 16:20 | 17:30 | |
| Heat 3 | | 13:00 | | 13:45 | | | 14:10 |
| | | 13:40 | | 13:55 | | | 15:00 |
| Heat 4 | | | 14:05 | 14:45 | | | 15:00 |
| | | | 14:45 | 14:55 | | | 15:45 |

Table 1: Rough scheduling of 4 heats. We have entered start and end time. Cranes and cars are not included.

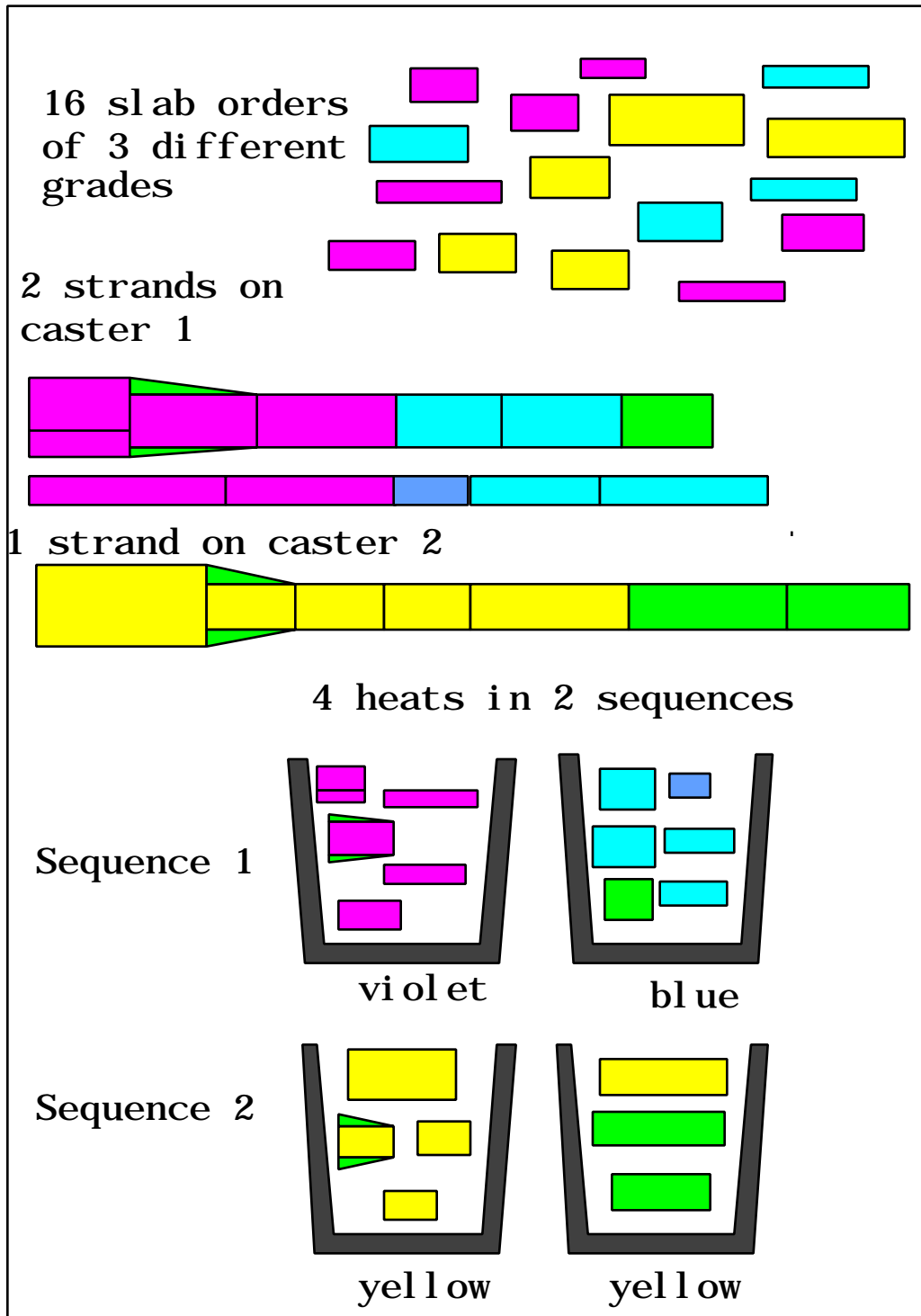


Fig.2: How to build up strands, heats, and sequences (simple example).

4 Short description of CCSP

CC SP (continuous caster sequence planner) is a production planning system for a steel plant (taking into account the process from the converter to the continuous slab caster). The planning horizon for "detailed (on line) planning" is one day.

The system was developed using object oriented methods, C++, and the libraries ILOG SOLVER 3.0 and ILOG SCHEDULER 2.0.

There are two mechanisms for adapting the system to a specific steel plant (see Fig.3).

1. The design of the steel plant, certain strategies, the different constraints, and the attributes of the planning objects are specified in a database.
2. A "strategy framework" can be used for implementing steel plant specific "expert knowledge".

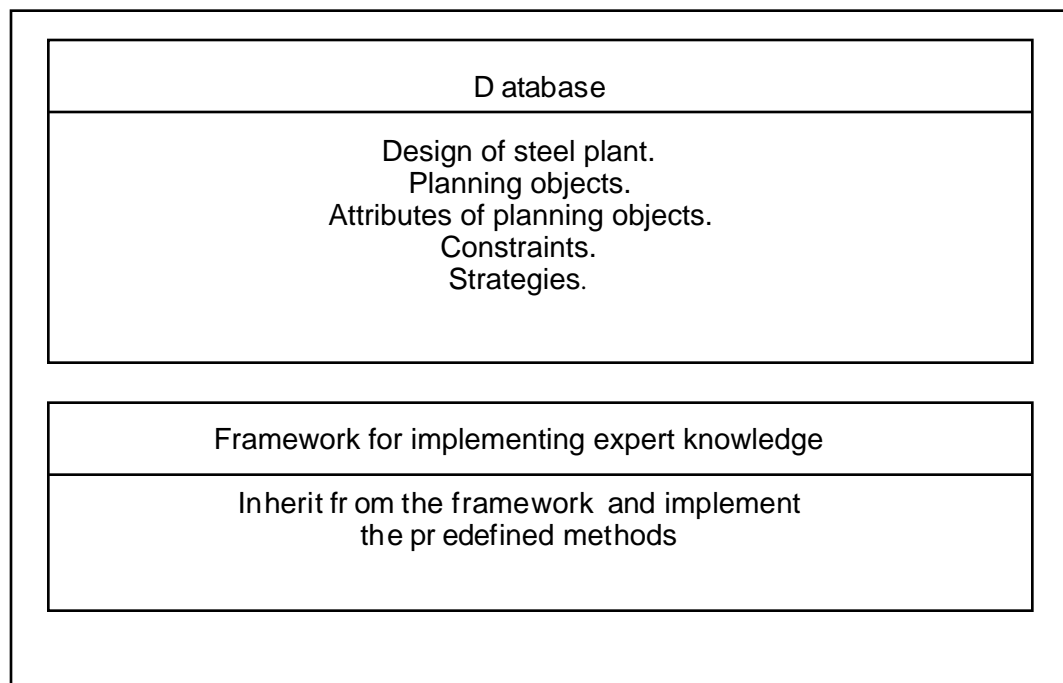


Fig.3: The two components which give flexibility to CCSP.

5 Mapping the steel plant to an object structure

In the following we will discuss those five modules of our system where we have used ILOG (see Fig.4).

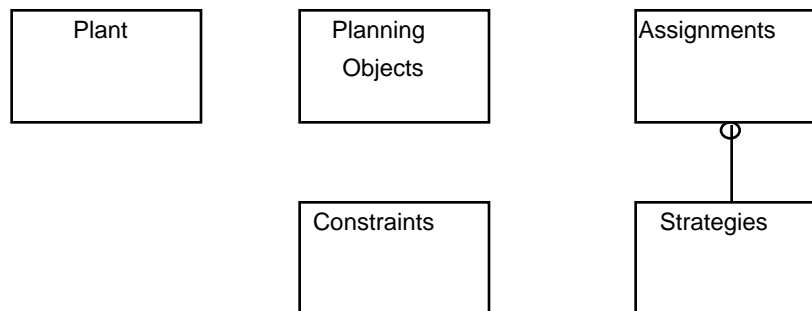


Fig.4: The five modules of CCSP where ILOG was used.

- **Plant:** Contains the scheduling part of CCSP. Resources and activities can be entered here.
- **Planning Objects:** Slabs, heats, and sequences are examples for planning objects.
- **Constraint:** The attributes of the planning objects are related by constraints to each other.
- **Assignments:** Planning objects have to be assigned to each other, e.g. slabs have to be assigned to heats, heats to sequences, and sequences to casters. The module Assignments uses the module Strategies.
- **Strategies:** Here we fix how the assignments are built up.

We show where we have used the ILOG class libraries, and where we had problems in doing so. ILOG classes are greyed (Remark: This is an improper use of the Booc h notation, where metaclasses are greyed).

5.1 Plant

This module is responsible for rough scheduling. The class `PlantHandler` (see Fig.5) has two containers. In the first one you can find all steel plant resources (caster, converter, secondary metallurgy equipment, ...) and in the second one the possible routings for all production grades. `Routing` itself has a container in which all activities are stored. `SteelPlantResource` inherits from the ILOG class `ILCResource`.

The `PlantHandler` has a method `trySchedule(...)`. If scheduling is possible `TRUE` is returned, otherwise `FALSE`. The use of ILOG was quite natural here.

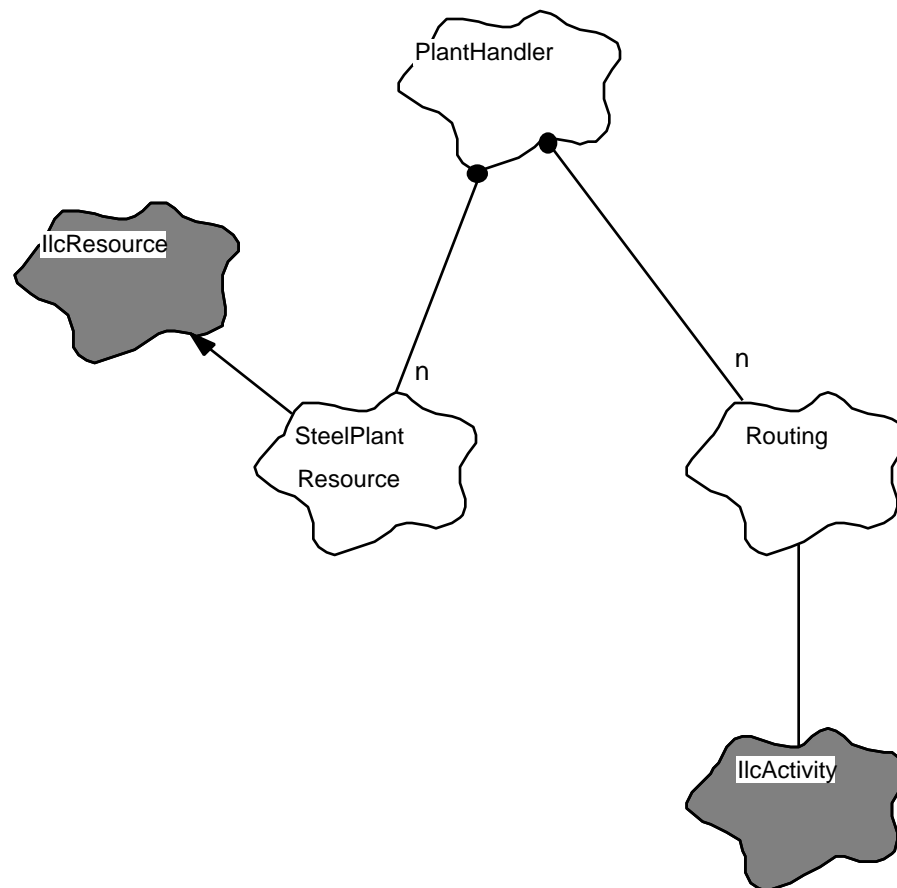


Fig.5: The `PlantHandler` is responsible for rough scheduling.

5.2 Planning Objects

In Fig.6 we have displayed the structure of our planning objects. The attributes are read from a database.

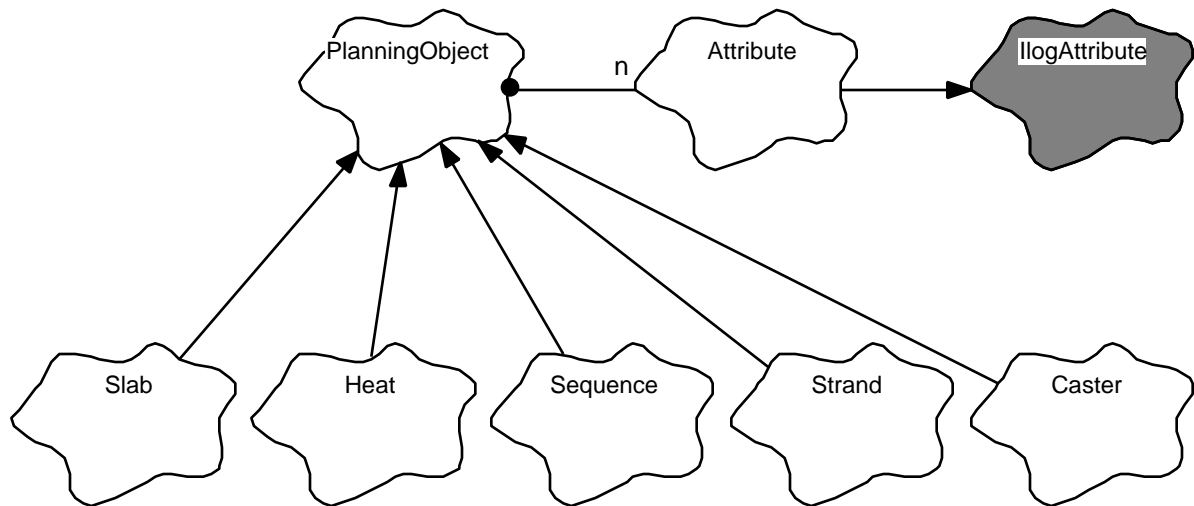


Fig.6: The planning objects used in CC SP.

A Slab could have the attributes {length, thickness, weight, productionGrade, ...} - the attribute productionGrade contains information about chemical composition and physical properties.

A Caster could have the attributes {casterId, castingStartTime, castingEndTime, noOfStrands, minimumStrandWidth, maximumStrandWidth, ...}.

Slabs are collected in Heats. A Heat could have the attributes {weight, noOfSlabs, castingStartTime, castingEndTime, ...}

5.3 Constraints

We have two fundamental types of constraints, i.e. Constructor Constraint(s) and Assignment Constraint(s), see Fig.7.

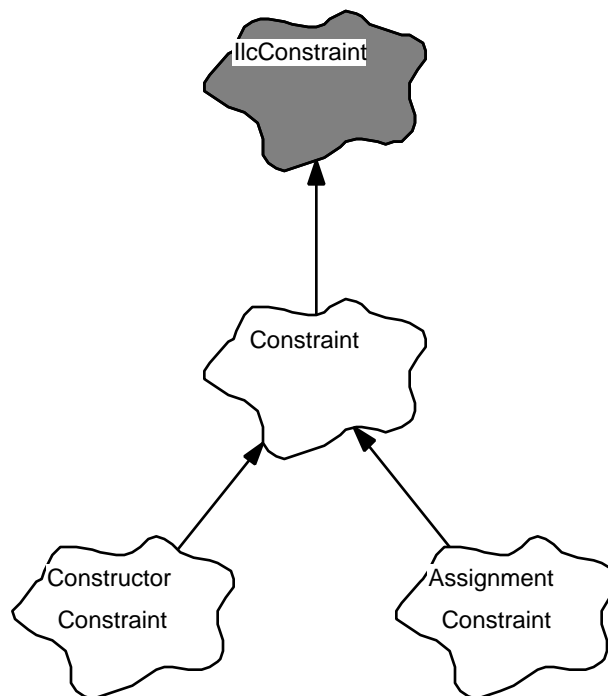


Fig.7: Constraints used in CCSP.

The constraints are read from a database. They look like follows:

Constructor Constraints:

- $\text{Slab.weight} = \text{Slab.length} * \text{Slab.thickness} * \text{Slab.width} * \text{Slab.density}$.
- $\text{Heat.noOfSlabs} > 10$
- $\text{Heat.weight} = 220 \text{ t}$

Assignment Constraints:

- For the first Heat on a Caster : $\text{Heat.castingStartTime} = \text{Caster.castingStartTime}$
- If the first three Heats fulfill $\text{Heat.X-content} \geq 0.2$, then the following Heats should fulfill $\text{Heat.X-content} < 0.2$.

5.4 Assignments and strategies

In Fig.9 you can find a quite simplified situation. We have displayed a set of slabs, heats, and casters, respectively. Building up a production plan is equivalent to performing the following two steps:

- Build up the assignments between these sets.
- Instantiate the domains of the attributes (i.e. of slab::grade, slab::length, heat::grade, heat::rank, ...).

The slabs have to be assigned to heats, and the heats have to be assigned to casters. The number of slabs and the number of casters are fixed, however, the number of heats is unknown. The Assignment Framework is a general framework which is capable of building up these assignments by using strategies, see Fig.8. The quality of the result depends very much on the quality of these strategies. Moreover, they influence very much the time and memory behaviour of ILOG.

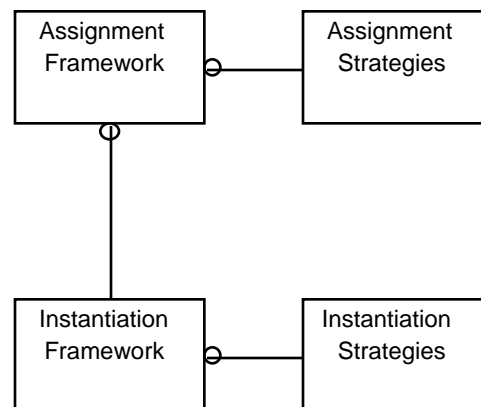


Fig.8: Assignment framework and some use relationships.

In an early version of our program we first built up all assignments; simultaneously all constraints became posted. Then we started to instantiate the attributes. However, two problems arose:

- The instantiation process takes too much time and memory.
- As soon as non linear constraints are included we have to fight with inconsistencies, i.e. ILOG does not recognize that some of the posted constraints cannot be fulfilled. (Remark: To use the ILOG method `IlcSolveBounds()` is not an option here, because of performance problems).

As a consequence we started to combine the two steps assigning and instantiating. However, starting with the instantiation process too early diminishes the quality of the planning results. It took us quite some time to find a solution for this problem which at least solved the inconsistency problem. But we still had performance problems. For solving these we had to adapt the instantiation process to our specific situation.

In summary we can say that the ILOG class libraries were quite useful for solving our problems. However, the use of ILOG SOLVER is quite non trivial. Many problems arose which we did not expect, and it took us quite some time to solve these problems.

Grafikname:
Erstellt in:
Erstellt am:

6 Portability Problems

We have developed our system on a HP UNIX workstation (32 bit architecture, 128MB main memory). Changing to a DEC UNIX workstation (64 bit architecture, 256MB main memory) was a major problem. Due to the different architecture we expected that our program would need about two times as much memory on the DEC than on the HP. However, this factor was between 3 and 5. Especially the ILOG solver heap grew excessively.