ERGONOMIC DESIGN FOR COOPERATIVE SCHEDULING SYSTEM

L. Haudot [1], P. Lopez [2], P. Esquirol [2 & 3], M. Sicard [1]

[1] DASSAULT AVIATION – 17, avenue Didier Daurat – 31701 Blagnac Cedex
phone : 33 – 61 77 41 84 – fax : 33 – 61 77 41 77 – e_mail : sicard@dassault–avion.fr

[2] LAAS du CNRS – 7, avenue du Colonel Roche – 31077 Toulouse Cedex
phone : 33 – 61 33 62 98 – fax : 33 – 61 55 35 77 – e_mail : lopez@laas.fr

[3] INSA Toulouse – Complexe scientifique de Rangueil – 31077 Toulouse Cedex
phone : 33 – 61 33 63 13 – e_mail : esquirol@laas.fr

Abstract : *This paper presents a study relative to the design of cooperative systems used for manufacturing scheduling. The study is based on the converge of the top down approach to functional design and the bottom up approach to the cognitive design of human–computer interaction (HCI). The top down approach applies a constraint propagation technique that aims to provide a set of solutions for decision–making. The cognitive approach applies knowledge acquisition techniques to better analyse the task and thus identify the user's needs. The result of this research was the production of two mock ups directly linked to the experimental pilot site: one for decision support used to arrange parts on a metal sheet before the first operation of cutting ; the other offers a cooperative environment for scheduling the remaining operations.*

Résumé : *Cet article présente une étude relative à la conception de systèmes coopératifs pour l'ordonnancement de production. L'étude s'appuie sur la convergence d'une approche descendante de conception fonctionnelle et d'une approche ascendante cognitive de conception d'interface homme–ordinateur (IHO). L'approche descendante applique des techniques d'analyse et de propagation de contraintes dans le but d'offrir un ensemble de solutions pour la prise de décision. L'approche cognitive utilise des techniques d'acquisition de connaissances afin de mieux analyser la tâche étudiée et identifier les besoins des utilisateurs. Ce travail a abouti à la réalisation de deux maquettes liées à un site pilote expérimental : l'une concerne une aide à la décision pour l'agencement de pièces sur une tôle avant leur découpe, l'autre propose un environnement coopératif pour l'ordonnancement des opérations suivantes.*

Key words : *Cooperation, Knowledge acquisition, Human–Computer Interaction, Scheduling, Constraint logic programming.*

Mots clés : *Coopération, Acquisition de connaissance, Interaction Homme–Ordinateur, Ordonnancement, Programmation logique avec contraintes*

1. INTRODUCTION

In this paper, we will present the results of the SCOOP project (Système COopératif pour l'Ordonnancement de Production) started with the MESR, the result of a collaboration between the LAAS – CNRS, Dassault Aviation and EURISCO [Haudot & al. 93]. This research is in keeping with a thesis carried out in the department of Artificial Intelligence and Advanced Computer Science at Dassault Aviation.

The subject that is of interest to us here is largely that of **cooperation.** A cooperative system is an organization where man and computer are seen as two interacting cognitive poles carrying out a task [Falzon 92]. Starting at the design phase, interaction modes with the user are taken into account in the architecture design of the system. The need for man–computer cooperation becomes more and more important with the growing development of *groupwares*, computing systems with which a group of people work [Ellis & al. 91].

It is very difficult (almost impossible) to elicit all the knowledge relative to a scheduling problem. Certain problems remain in the domain of implicit expertise, and for reasons or flexibility and reactivity, nothing replaces human decision making.

2. COOPERATION AND SCHEDULING

The scope of the SCOOP project concerns a shop that deals with problems linked to manufacturing. Schematically, a scheduling problem involves defining who should do what and when. It can also be defined as a problem where starting and control of operations should be assured during all manufacturing steps.

Scheduling problems [Baker 74] are often described as difficult, because of combinatory explosion. In order to give an outline of this phenomenon, let us take the following examples : for a problem consisting of 3 jobs to be done by 2 machines, there are 36 possible exchanges, corresponding to the various possible sequences of the jobs, and allocations of the jobs to the machines; for a problem involving 5 jobs and 5 machines the number of exchanges is in the neighborhood of 25 billions !

Operational Research attempts to find solutions to these combinatory problems. Most of the time, a good tradeoff between the quality of the solution and the computing time consists in designing specific heuristics .

The result is a schedule that represents a forecasted plan for carrying out the various tasks, which can serve as a guideline for decision–making. In the best of cases this plan provides only very partial information on the degrees of freedom (in the form of margins) which is available to react to disturbances, without forgetting the global objectives. Thus, the initial plan can no longer be followed because of its inflexibility. It can be periodically recomputed, taking into account the current state of the production system. In this case, a preliminary characterization of solutions avoids having to recalculate systematically a complete solution when the initial plan is partially disrupted.

One characteristic of our application is to deal with limited series production, aided by low automated processes ; in this context, the human factor is extremely important. For the moment, very few scheduling systems approach the problem from the cooperation point of view. When an optimal resolution approach is chosen, the diversity, and sometimes the antagonism of the formulated criterion,

the multitude and heterogeneity of the constraints and the implicit knowledge of deciders, forces the consideration of a much too simplified model compared to reality. Expert systems attempt to avoid these shortcomings, but while these tools are developed in close connection with experts, future users are rarely consulted, which results in an interaction model unadapted to the users' cognitive model.

Two systems should however be mentioned. ORDO is a software for real–time scheduling, developed by Cabinet Villaumié S.A. The main idea of ORDO is to offer not only one schedule, but a family of schedules, in the form of a totally ordered sequence of groups of exchangeable tasks [Le Gall & Roubellat 89]. ORDO is currently established in 40 industrial sites, in various domains of application. The SCHEPLAN [Numao & Morishita 91] software provides a cooperative environment for creating production plans in the steel industry. Constraint resolution procedures and expert rules enable the user to modify the plan by direct manipulation through an interactive interface. The modes of interaction with the user are realy taken into account in the design phase of the system ; its architecture is defined around a constraint solver.

Given the above remarks and criticisms, the two following points will be developed in the section dealing with our design methodologies :

– part of our efforts were concentrated on an approach deriving necessary conditions for the feasibility of solutions to scheduling problems. This is the so–called **constraint approach**. It is based on the rigorous analysis of a problem, independent of the way in which decisions are made. This allows an incremental resolution, entirely controllable by the user. This approach favours the flexibility of the system, necessary for a cooperative tool, while guaranteering consistency of the decisions given the formulated constraints.

– in order to study cognitive processes for problem resolution, a large part of this work concerns the **knowledge acquisition** of the users who work with the actual scheduling systems. It is based on an *in situ* analysis of the scheduling task, on interviews and on writing meetings (brainwritings).

3. METHODOLOGIES

We will now deal in detail with the methodologies previously mentioned.

3.1. The constraint approach

3.1.1 The Constraint–Based Analysis (CBA) of scheduling problems [Erschler & al. 80].

As opposed to generating a unique solution relative to some optimization criteria, the goal here is to simplify the resolution by the active use of the constraints. To do this, a set of deductive rules makes it possible to reveal some vital properties for all solutions. This process aims to reduce the combinatorial of a problem by eliminating, as much as possible and as early as possible, incompatible decisions with respect to the constraints. In this way, the domain of certain decision variables can be drastically reduced before any decision is made about their value. This procedure also enables the detection of some inconsistencies between different types of constraints. Even though this analysis is independent from the chosen method of resolution, it lends itself to an approach that favours full user–guided resolution. Once done, it can be followed by a phase where the user can makes decisions according to his expert knowledge.

Constraint–based analysis can be viewed as an instanciation of more general problematics, the **constraint approach for problem solving**. The previous has given rise not only to theoritical research, classifiable under the name of Constraint Satisfaction Problems [Mackworth 77], but also to the creation of **Constraint Logic Programming** languages (CLP), e.g., PrologIII, Charme or CHIP. The latter was used for this study, and will be briefly presented in the next paragraph.

Combining the CBA principles and CLP languages may have some advantages. From the efficiency point of view, a real improvement has been made, because of the dedicated nature of these languages, compared to realizations programmed in general languages such as C, Pascal or ADA. Conversely, some kinds of reasoning, specifically pertaining to scheduling constraints, suggest some improvements of the internal mechanisms of constraints management in CLP languages. A strong convergence exists between these two paradigms with the possibilitity of mutual enriching.

3.1.2. The CHIP language

To implement our system, we chose the CHIP (Constraint Handling In Prolog) CLP language [Dincbas & al. 90]. Posting constraints in an *a priori* way can cause reducing modifications on the variables domain. This mechanism of implicit propagation of constraints enables the pruning of the search tree during the resolution. Indeed, as soon as a decision is made (variable instanciation), the set of constraints in which it appears is considered, not only to simplify its expression but also to deduce new restrictions on the domains of the remaining free variables. This active way of managing constraints can be opposed to the classical naïve principle of constraints checking, the last one requires a complete instanciation of variables ("generate–and–test" principle). Amongst the main reasons which led us to prefer the CHIP language before than others, we would point out :

– the strong structural relationship between CBA rules and CLP, that enables us to re–use some of the previous software packages, written in standard Prolog,

– the specialized nature of CLP languages,

– the efficiency of constraint propagation mechanisms on domain variables, and particularly the possibility of managing discontinuous ranges,

– the advanced graphical primitives for building interfaces.

3.1.3. The constraint–based approach for cooperation

Thanks to CBA rules efficiently implemented in CLP, our system is able to deduce a more and more reduced decision space, progressively as decisions are made by the user. The constraint–based approach gives the system the power to react to the users decisions, and turns him back their logical consequences on the admissibility of the remaining problem. It is a first form of cooperative behavior, that eliminates as early and as much as possible any decision which might be inconsistent with the constraints. A second form of cooperation aims at favouring the decision phase, by selecting and sorting the remaining choices in increasing order of preference. This aid is more constructive, and the user keeps the control and may reject any suggestion. Selecting and sorting criteria can be updated at any time by the user, becoming relevant only when a sufficient level of resolution is reached. One of these criteria considers the temporal characteristics of the cutting operations. To implement it, we have used some techniques of *data analysis*.

3.1.4. The data analysis [Lerman 81, Erschler & Pradin 92]

In order to favour the decision making, the interface presents to the user the list of cutting operations which are not already allocated to any metal sheet, and mentions for each of them a list of preferences for the possible arrangements.

From the presentation point of view, the user can activate various filters (sheet type, max. or min. area, due date, ...), in order to focus on a limited set of operations. He can also sort the list of operations with the help of various combinable criterion. One of them is based on the limit time of processing (allowed time window). A "distance" value is computed between each unscheduled operation and each sheet : this value is infinite in the case where it is impossible to allocate the operation on the sheet due to one arrangement constraint (area, type, time window). In less extreme cases, the value is proportional to :

– the relative distance between the time window of the operation and the time window of the aggregated set of operations already arranged on the sheet,

– the time margin lost by allocating the operation to the sheet.

The goal of this sorting criteria is to reveal the allocation decisions that will be less constraining from a time flexibility point of view. It has two effects. When a solution is found according to this criteria, it is more robust with regards to the time disturbances that may occur during processing (the margins being larger compared to an ordinary solution). The resolution effort is diminished since the risk of failing (due to the violation of temporal constraints) decreases when margins are larger.

3.1.5. Conclusion

The decision support given by the system comes from the combination of, on one hand, the analysis before decision (by selection and classification of the most appropriated decisions), and on the other hand, the analysis after decision (by infering the consequences and eliminating wrong values for the remaining variables). The main objective of our approach is to trade on these two kinds of analysis not for an automated resolution but for an incremental resolution controlled by the user. Such an approach has been judged more realistic particularly because the number of daily decisions was low on the pilot site (despite a high number of constraints). Thus the global resolution time remains short.

3.2. Knowledge acquisition techniques.

The knowledge acquisition phase aims to help us to understand the task and to take into account the needs of the users.

This phase is defined in three stages :

– knowledge collection,

– knowledge structure,

– concepts analysis and generation of design.

3.2.1. Knowledge collection.

The first task of the cognitive engineer consists in getting used to the concepts and the vocabulary of the domain he wishes to study, with the intention of better understanding his futur interlocutors. After this learning phase, different methods can be used to collect knowledge :

– observation of the users *in situ* by the cognitive engineer,

– recording the user's interactions with their initial systems, using sensors for example,

– direct interview with the users,

– classical meetings / discussions (brainstorming),

– writing meetings (brainwriting).

In the scope of this research, we have mainly carried out a brainwriting campaign composed of users and designers. The designers give their knowledge about the technological possibilities and the methodologies, the users bring their needs concerning the scheduling tools.

The brainwriting is a collective knowledge acquisition technique. Each participant replies by writing, over a grid (see figure 1) to a question asked by the cognitive engineer at the beginning of the session. So that each person can react to each idea expressed, the grids of answers are exchanged, during a same session (for example, using a circular exchange). This process is repeated until all the participants have read and eventually written over each grid.

ideas numbers	**refers numbers**	**opinion ? (Yes/No)**	**ideas and remarks**
1	–	–	If the model used is formal the system could take decisions.
2	1	Yes	In the opposed case operator must take decision from suggestions computed by the system.

fig.1 Example of grid

3.2.2. Knowledge structure

Once collected, information is classified and structured within a conceptual tree of specialization. This tree includes both interaction and functional concepts. Different methods may be used to help the cognitive engineer in this classification phase (types–based classification, attributes–based classification, ...). Nevertheless, the empirical method remains the most frequently used and the structuration of the tree depends on the know–how of the cognitive engineer. Some examples of classes identified during our research on the topic of scheduling (corresponding instances are given in parenthesis) are given below :

– user type (production manager, shop–floor manager, worker, ...),

– information type (tasks, resources, products, routings, inventories, loads, dates, delays, ...),

– display modes (graphs, arrays, curves, bargraphs, charts, ...),

– actions (add, delete, modify, allocate, ...),

Note that these classes are generic, and that only their instances can be specific to the scheduling domain.

3.2.3. Knowledge analysis

The analysis is built by crossing ideas in order to determine the links between concepts. The semantics of these links is dependent on the goals on the cognitive engineer, as for example :

– preponderance relations : idea i is more important than idea j,

– causal relations : idea i implies idea j,

– simultaneity relations : idea i and idea j must be realized simultaneously,

– compatibility relations : idea i and idea j are compatible, ...

To obtain these relations, a matrix that crosses some given subsets of collected ideas is presented to each participant of the knowledge collection phase. Each one must give his opinion about the link between the idea in column i and the idea in row j, through a value written in the slot Cij [Boy 91]. These matrixes are then processed to reveal agreement and disagreement points [Shaw & Gaines 88]. With this aim in mind, it is possible for a given semantic to build consensus matrixes, by computing the sum –weighted or not– of the slots cij over the set of individual matrixes. This sum is then compared with 2 reference thresholds, t1 and t2 :

– if Σ cij $\leq$ t1, a consensus exists about the irrelevance of the link (the link must be rejected),

– if Σ cij $\geq$ t2, a consensus exists about the relevance of the link (a rule is generated),

– if t1 $\leq \Sigma$ cij < t2, a disagreement exists about the the relevance of he link. A new session of brainwriting can start on this subject, to reach a consensus. It should be noted that one of the possible causes of disagreement can be found in the existence of several different terminologies for the same concept amongst the participants.

Below is an example of analysis carried out from crossing two classes of ideas : information types and display modes. The matrix on the left represents the answer–matrix of one user, and the matrix on the right one represents the consensus matrix built from 5 answer–matrixes.

Crossing : information types / display modes Number of participants: 5

Sémantics : Are display mode **B** and information type **A** compatible **?** (yes = 1, no = 0)

M1 : answer–matrix of a participant

A / B	routings	tasks	stocks	resources
graphs	1	1	0	0
bargraphs	0	0	1	1
curves	0	0	1	1
plans	1	1	0	0
lists	1	1	0	1

cij = 0 –> no

cij = 1 –> yes

$\mathcal{M}_c = \Sigma$ Mk : consensus matrix

A / B	routings	tasks	stocks	resources
graphs	5	3	0	1
bargraphs	0	0	5	4
curves	1	0	4	5
plans	2	5	1	0
lists	4	5	0	4

Σ cij $\leq$ 1 –> incompatibility (exclusion rule)

Σ cij $\geq$ 4 –> incompatibility (acceptance rule)

1 < Σ cij < 4 –> disagreement

fig.2 Two–dimensions analysis

Several knowledge aquisition tools have privileged the form of bi–dimensional matrixes for knowledge representation [Lafrance 86, Bradshaw & al. 93]. Nevertheless, in order to acquire contextual information, n–dimensional crossings can be imagined. Below are two examples of 3 and 4 dimensions, that are relevant to the scheduling domain :

– information type / display modes / actions,

– information type / display modes / actions / interaction modes.

After this processing phase, some design rules are generated (exclusion or relevance links), in addition to disagreement points that will be eventually solved by the cognitive engineer during new knowledge acquisition sessions (this process continues until all disagreements are solved).

A first brainwriting permits to extracting two major categories of problems : (1) the user needs and (2) the technological feasibilities. Brainwritings have been carried out again for each of these categories, in order to precise some particular concepts.

Such a knowledge acquisition procedure led us to generate specific rules in the fields of cooperation and interaction.

Cooperation

The following points have been raised for the establisment of cooperation processes :

– the possibility to question the validity of already made decisions, facing the dynamic context in the shop–floor,

– the anticipation necessity for making decision in advance. Indeed, at each step of the decision–making, the tool has to exhibit the most robust choices so as to limit the backtrackings as much as possible.

Moreover, the decision sharing remains a preponderant factor :

– the machine (and specific heuristics) solve the problems whether they are well–modeled and of great complexity,

– the man makes decision for problems not well formalized.

Interaction

Below is a first extract of the generated interaction rules which should be improved.

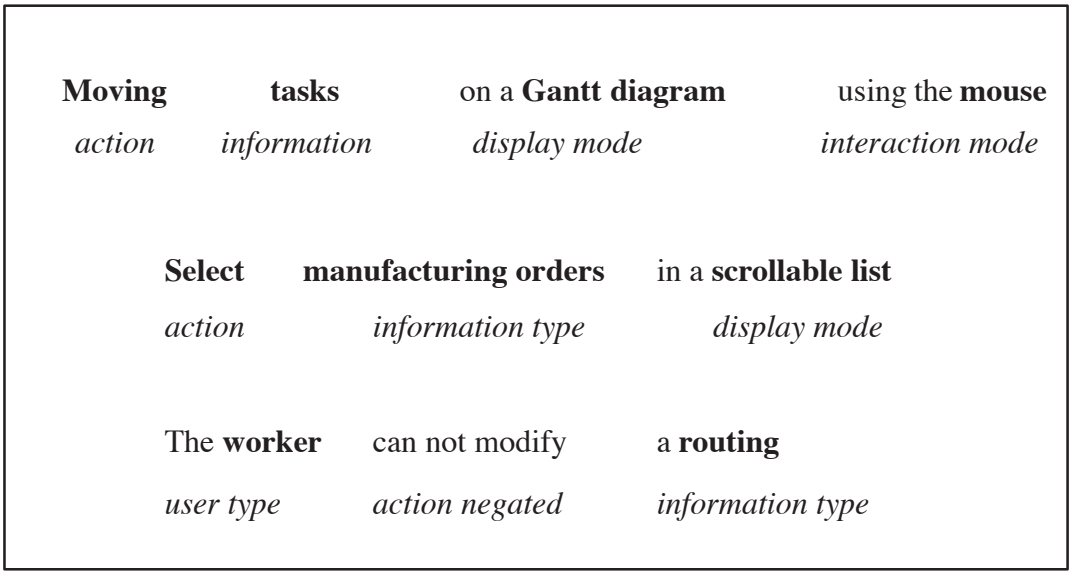


fig.3 Examples of interaction rules

3.2.5. Discussion

This methodology has advantages and drawbacks. The great advantage of the brainwriting technique is avoiding that participants be influenced or interrupted by strong personalities, as it is often the case in classical meetings. However, the spontaneaous nature of the latter disappears and the convergence of opinions is very slow or even unrealizable.

The formalization phase is also a critical point. The classification of information is indeed a weighty task, that could be aided by tools, such as automated indexation systems. Nevertheless, the role of the cognitive engineer is still of great importance.

Finally, any kind of matrix is not efficient during the analysis phase. A real difficulty exists when choosing the sets of ideas to be crossed.

4. PROTOTYPING

We have applied cooperation and interaction rules as a design support methodology for cooperative systems. Our problem was the scheduling of the cambered metal sheet shop of the Argenteuil plant of the Dassault Aviation company. During the implementation, some general ergonomics principles [Afnor 88, Coutaz 90] like compatibility, guidance, homogeneity, flexibility, control, concision or errors handling are also respected.

This shop is in charge of a limited series production, which implies a great flexibility in management and the necessity to react on a very short term basis ; production processes have a low level of automation (many operations are done manually). For these reasons, the role of human decision–making is then preponderant.

Routings can be different for each product. Parts are cut out of thin metal sheets, then formed with a hydraulic pressing machine. Each part is characterized by its reference number, its routing, its class and its due date.

On the first station (cutting station), the problem is to determine the sets of parts that will be cut out simultaneously on each metal sheet. The built sets must satisfy constraints such as class constraints,

geometric imbrication constraints, filling rate constraints, and also time constraints coming from the delivery date associated to each part. This phase governs the release of the parts in production and strongly influences the scheduling of the next operations. At present, although they are automatically generated, the sets of parts are almost systematically modified then finally validated by the operator. These alterations are necessary in order to integrate the current context of the shop (rubbishes, urgent orders ...) that can be done by the existing system. Besides this shortage, we can also highlight the absence of flexibility (due to a batch–oriented generation), and the absence of indicators which could facilitate decision–making. Consequently, there is a strong need for a **cooperative decision support system**.

Once solved, the problem of building the sets of parts on the first work station, a second system is needed by the shop floor manager who achieves the complete scheduling of the operations following the cutting out stage.

As for the previous example, this system can not take into account the whole context of production ; furthermore it has to be reactive. We have also developed a cooperative scheduling tool, used by the shop floor manager to build/modify interactively the solution.

Both systems are based on the same generic functioning principle which benefits from the methologies described above.

Figure.4 schematically shows the different functional units that we propose for a generic cooperative system in the domain of scheduling.

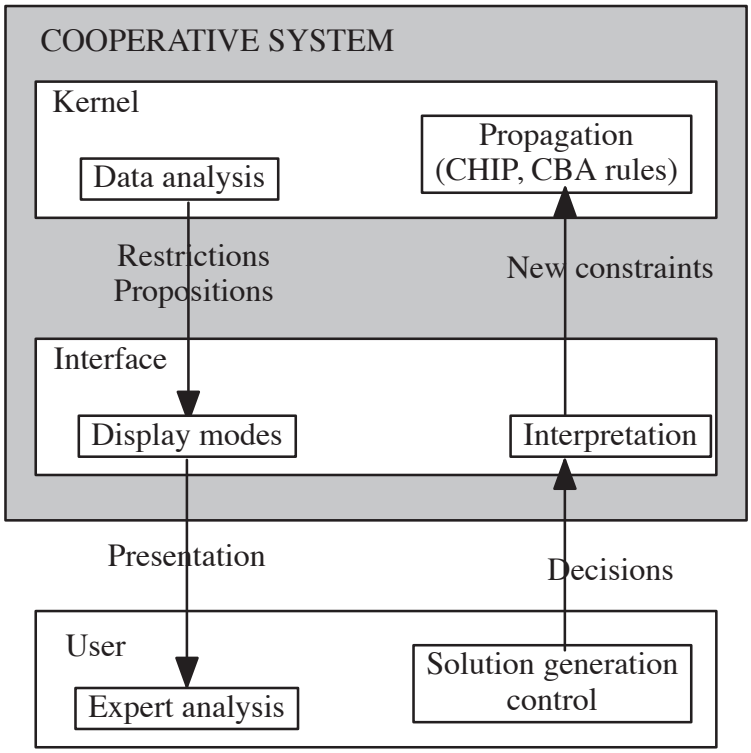


fig.4 Functional principles for cooperative scheduling system

4.1. The release package

Our prototype interactively builds the sets of parts to be cut out, from the list of production orders. This process is decomposed into two groups of functions :

– parts selection support functions,

– metal sheet filling support functions.

4.1.1. The parts selection support functions

For each part family, the system gives a graphical representation of the orders state (relative proportion of each ordered quantity and time staggering curves). From these representations, a subset of parts is extracted, using one or more criteria specified by operator (routing, area, delivery date and family). Focussing on this list, the operator can then begin to build the groups of parts.

4.1.2. The metal sheet filling support functions

The operator defines the parts to be arranged on the same metal sheet. The system verifies their compatibility with the constraints (type, area, geometry, delivery date). If the process fails, the cause is then displayed (insufficient remaining area, incompatible type, outrun delivery date, ...). The metal sheets currently being filled are graphically visualized, as well as their scheduling on the cutting machine. Updatings are made in real–time. At any time, the operator can go back on all previous decisions, no matter what order they have been taken in.

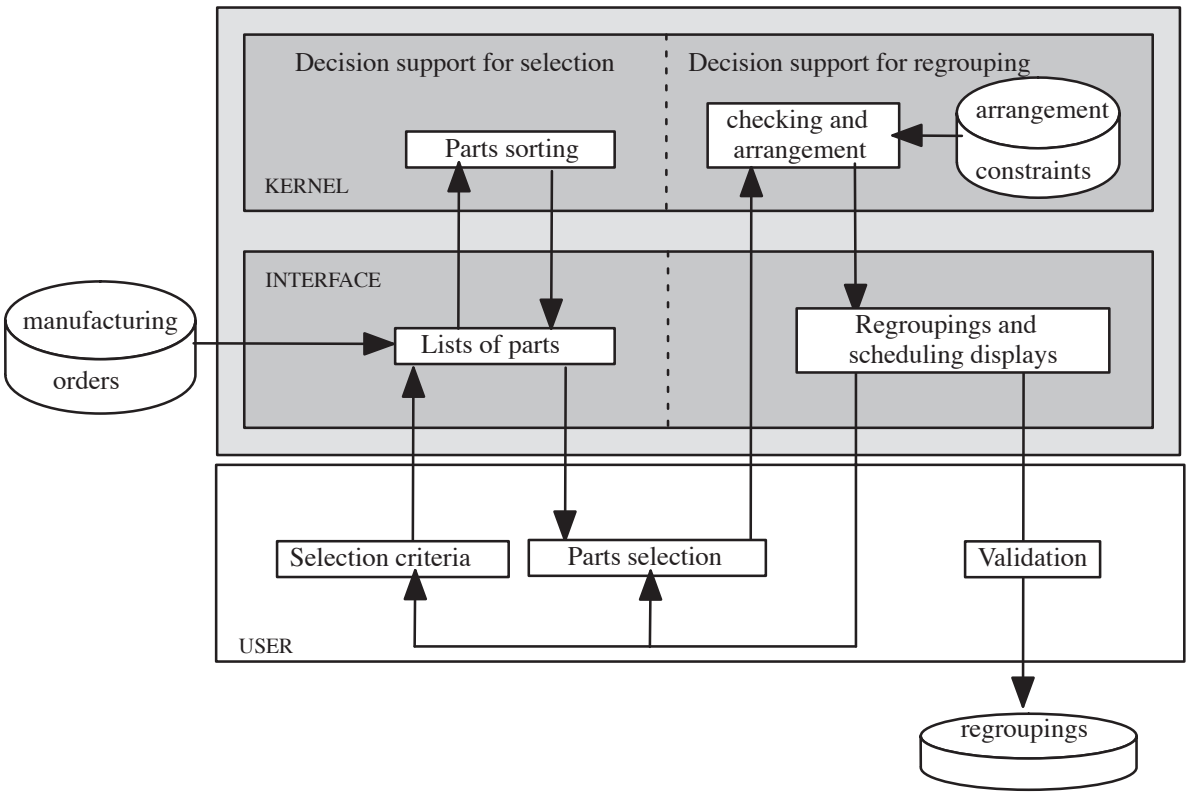


fig.5 Human–Computer cooperation for release

1 list of selected parts

2 regroupings building

3 cutting out operations planning

fig.6 Release system

4.2. The scheduling package

This supports the interactive design of the scheduling. Tasks are visualized with their margins on a time windows schedule. The shoop floor manager can move and resize them. He can also add new sequencing constraints between tasks. After each action, the system updates the time window of each task. Moreover, to make his decisions, he can refer to some indicators, such as the resources load indicator or the browser of precedence links between tasks. A global or step–by–step resolution mode can be fixed. As for the operator tool, the manager can at any time backtrack on his decisions.

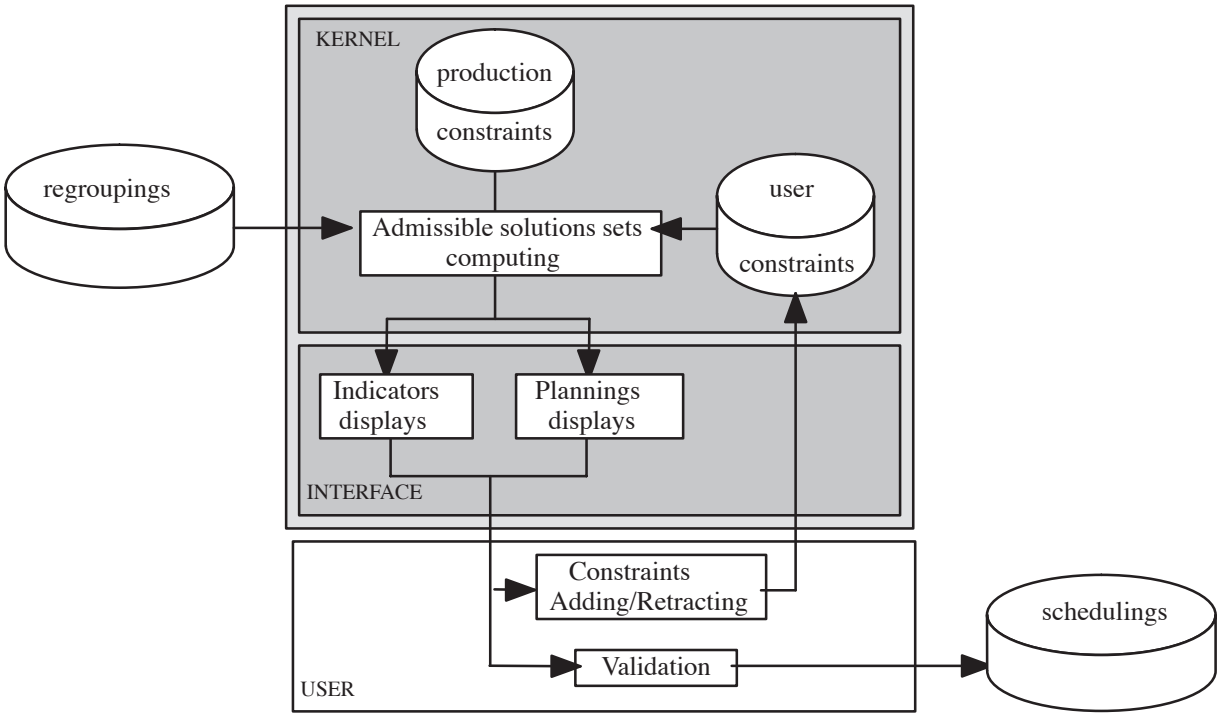
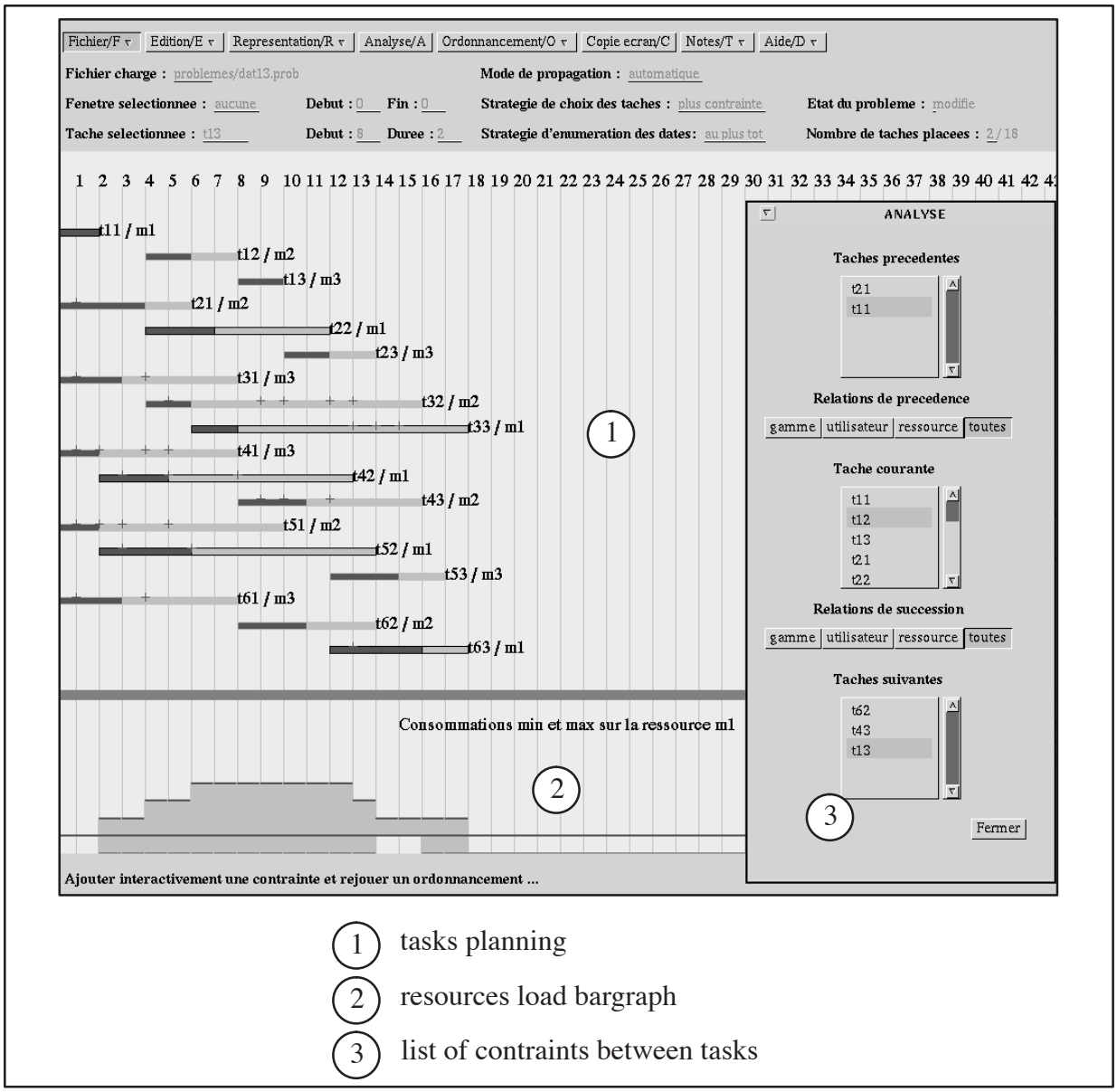


fig.7 Human–Computer cooperation for scheduling



1 tasks planning

2 resources load bargraph

3 list of contraints between tasks

fig.8 Scheduling system

5. CONCLUSION AND OPEN ISSUES

5.1. Validation

These two systems are today at a mock up level of realization and have to be validated by users. This validation phase belongs to the normal incremental cycle of prototyping, that will obviously imply some adjusting and innovating improvments.

5.2. Systems connexion

The two systems will have to be connected to achieve the complete scheduling of the shop. These groups of parts (output of the first tool) may be reconsidered either when the complete scheduling fails, or following some decision of the shop floor manager. The rejected groups will be then modified by the operator with the support by the global system (release and scheduling packages).

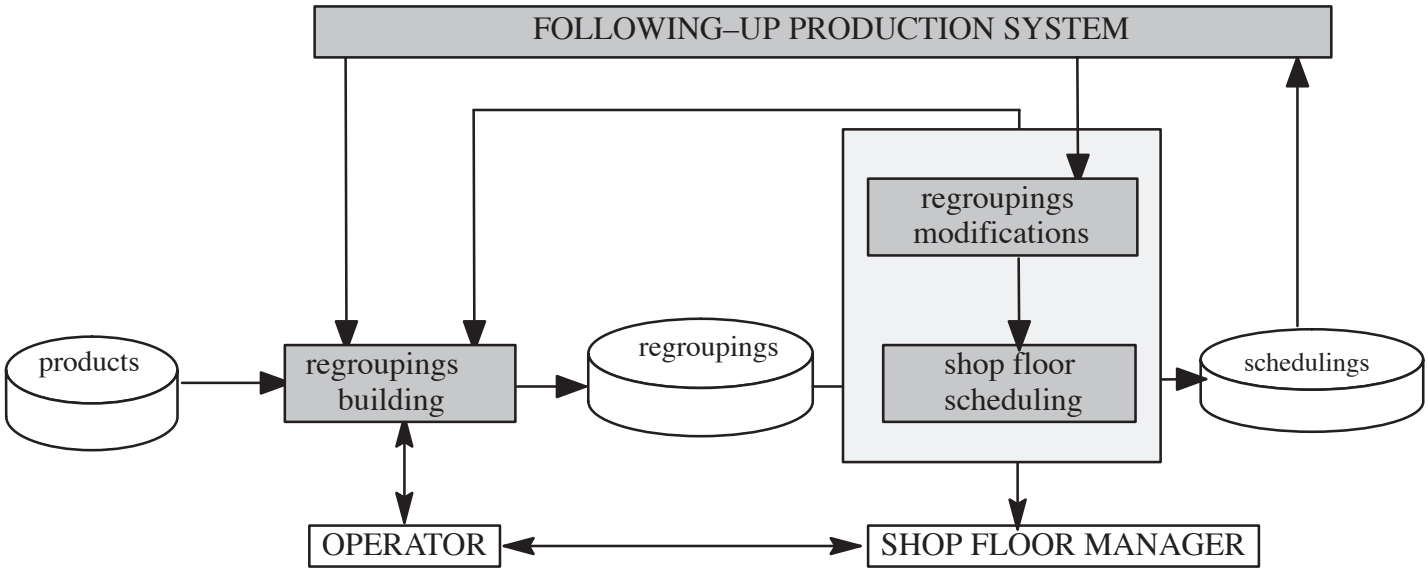


fig.9 Systems connexion

5.3. Genericity of the methodologies

The approaches developed in this paper seem to be well adapted to HCI design. Their genericity make them reusable for the design of dedicated tools in others domains.

This multi–disciplinary project has brought back some experience on the advantages and outcomings of the cooperation between three research teams with very different skill profiles. It has demonstrated how essential the expertise of the designing team is in the domain of knowledge acquisition.The development of new incremental tools for acquisition and prototyping seems to be very relevant for designing user–oriented software applications.

6. AKNOWLEDGEMENTS

We thank Guy Boy and Jeffrey Bradshaw from EURISCO for helping us during the knowledge acquisition phase. They initiate us to the different methodologies we adapted to the cooperative scheduling problem (not approached before in this way).

7. REFERENCES

– [Afnor 88] *Ergonomie et conception du dialogue Homme Ordinateur,* Fascicule de documentation Z 67–110 , janvier 1988,

– [Baker 74] K.R. Baker (1974), *Introduction to sequencing and scheduling*. John Wiley, New–York,

– [Boy 91], G. Boy (1991), *Intelligent Assistant Systems*, Academic Press : London,

– [Bradshaw & al. 93] J.M. Bradshaw, K.M. Ford, J.R. Adams–Weber, J.H. Boose (1993), *Beyond the repertoy grid : new approaches to constructivist knowledge acquisition tool development,* Knowledge acquisition as modeling, pp. 287–333, K.M. Ford & J.M. Bradshaw (Eds.), John Wiley : New–York,

– [Coutaz 90] J. Coutaz (1990), *Interfaces Homme Ordinateur : conception et réalisation,* Editions Bordas : Paris,

– [Dincbas & al. 90] M. Dincbas, H. Simonis, P. Van Hentenryck (1990), *Solving large combinatorial problems in logic programming*, Journal of logic programming, Vol 8(1–2),

– [Ellis & al. 91] C. Ellis, S. Gibbs, G. Rein (1991), *Groupware: Some issues and experiences,* Communications of the ACM, 34(1), pp.38–58,

– [Erschler & al. 80] J. Erschler, F. Roubellat, J.P. Vernhes (1980), *Characterizing the set of feasible sequences for n jobs to be carried out on a single machine,* European Journal of Operational Research, Vol 4(3), pp.189–94,

– [Erschler & Pradin 92] J. Erschler B. Pradin (1992), *Décomposition temporelle et problèmes d'ordonnancement*. Distancia'92, Rennes, 22–26 juin,

– [Falzon 92] P. Falzon (1992), *Vers des partenaires cognitifs,* Le courrier du CNRS, Vol 7, p.102

– [Haudot & al. 93] L. Haudot, M. Sicard, P. Esquirol, P. Lopez, G. Boy, J. Bradshaw (1993), *SCOOP : Système COopératif pour l' Ordonnancement de Production*, Rapport de fin de recherche MESR,

– [Lafrance 86] M. Lafrance (1986), *The knowledge acquisition grid : a method for knowledge engineers,* Knowledge acquisition for knowledge–based systems workshop, Banff, Canada, november,

– [Le Gall & Roubellat 89] A. Le Gall, F. Roubellat (1989), *A decision support system for real time production scheduling,* Third ORSA/TIMS conference on Flexible Manufacturing Systems: Operations Research Models and Applications, Cambridge, Massachussets, USA,

– [Lerman 81] I.C. Lerman (1981), *Classification et analyse ordinale des données*. Dunod,

– [Mackworth 77] A.K. Mackworth (1977), *Consistency in networks of relations,* Artificial Intelligence, Vol. 8, pp.99–118,

– [Numao & Morishita 91] M. Numao, S. Morishita (1991), *Cooperative scheduling and its application to steelmaking processes,* IEEE Transactions on Industrial Electronics, Vol 38(2), pp.150–155,

– [Shaw & Gaines 88] M. L. G. Shaw and B. R. Gaines (1988), *A methodology for recognizing consensus, correspondence, conflict and contrast in knowledge acquisition system,* 3rd AAAI–Sponsored Knowledge Acquisition for Knowledge–Based Systems Workshop, Banff