7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France

« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

CONCEPTION COLLABORATIVE NON SUPERVISEE A BASE DE CONTRAINTES

APPLICATION A LA CONCEPTION D'UN TRAIN D'ENGRENAGES

Pierre-Alain YVARS

LISMMA Supméca Paris
3 rue Fernand Hainaut
93407 saint Ouen cedex
payvars@supmeca.fr

RESUME *: Nous considérons dans ce papier, qu'un problème de conception routinière de type dimensionnement ou configuration de produit peut être modélisé sous la forme d'un CSP (Constraint Satisfaction Problem). La modélisation et la résolution de CSP a fait ses preuves dans le cadre d'une conception mono concepteur. Nous proposons d'étendre les fonctionnalités des CSP au cadre d'une conception multi concepteur d'un même artefact. Pour se faire, nous présentons les principes de fonctionnement d'un algorithme pour la gestion de décisions de conception en temps réel, s'appuyant sur un modèle du produit décrit sous la forme d'un CSP. Cet algorithme permet de minimiser le nombre de décisions de conception rejetées à un instant donné de la conception. L'algorithme compose le cœur d'un prototype de système de conception collaborative non supervisée à base de contraintes. Ce prototype a été testé sur le cas d'un dimensionnement simultané d'une transmission de puissance par engrenages.*

MOTS-CLES : *conception collaborative de produit, CSP, contraintes, détection de conflits.*

1. INTRODUCTION

Les systèmes d'aide à la conception en ingénierie mécanique ont évolué de la représentation purement géométrique du produit vers l'intégration des connaissances métier (Bernard, 2000).

Ainsi, (Chandrasekaran, 1990) propose une classification communément admise des problèmes de conception. Il distingue ainsi trois types de conception : la conception créative, innovatrice et routinière. Une conception créative devient innovatrice lorsque les fonctions du futur produit ont été définies et qu'il s'agit de décider des principes de fonctionnement et de les mettre en oeuvre. (Kota et Ward, 1991) parlent également de conception routinière pour l'étape de réalisation des choix parmi les composants disponibles et de dimensionnement de l'ensemble des éléments du produit.

Le cadre de nos travaux se situe dans le champ de la conception routinière qui occupe en moyenne 80 % de l'activité d'un concepteur. De nombreuses techniques peuvent être utilisé pour l'exploration d'espace de conception, tant pour la recherche de solutions que pour leur optimisation éventuelle.

Nous pensons que ces techniques peuvent être rangées en trois catégories :

- Approche constructive : Ou partant d'un point de l'espace de conception où l'ensemble des variables de conception est libre, nous cheminons de manière incrémentale vers une solution admissible au fur et à mesure de l'instanciation des variables du problème. En pratique, ce processus est rarement linéaire et est sujet à de nombreux aller et retour. La représentation par graphe d'états est une bonne manière de modéliser ce type d'approche.

- Approche par amélioration de solutions : Ou partant d'une solution approximative de notre problème et en procédant à une suite de modifications locales on arrive à une solution admissible. Souvent cette solution est localement satisfaisante et peut se trouver piégé dans un puit de potentiel. Différentes méthodes permettent de sauter ces puits de potentiels. Certaines partent d'une seule solution initiale comme (Kirkpatrick et al, 1983) avec le recuit simulé ou (Cvijovic et Klinowski, 1995) avec la recherche tabou. D'autres partent d'un ensemble de solutions potentiels comme (Barricelli et all, 1954) et (Falkenauer 1997) avec les algorithmes génétiques ou bien (Clerc,2005) avec l'optimisation par essaim de particules ou (Dorigo ,1992) avec les colonies de fourmis mis à jour dans (Dorigo et Stützle, 2004).

- Approche de type Case Based Reasoning (CBR) et data mining : Ou l'on va chercher dans une base de données ou une base de cas

existante une solution proche du cahier des charges que doit satisfaire notre artefact comme dans (Sveda , 2007).

Depuis quelques années, nous nous sommes intéressés à l'utilisation des techniques de programmation par contraintes pour la résolution de problèmes de conception. Nous avons entre autre mis au point des méta modèles adaptés pour la modélisation de produit et de raisonnement de conception en vue d'une résolution par contraintes (Yvars, 2001). Ces travaux portaient essentiellement sur une résolution mono concepteur et nous avons par la suite cherché à étendre tout cela à un environnement multi concepteur.

Nous proposons dans ce papier d'examiner l'apport des techniques de modélisation et de résolution de CSP pour la résolution coopérative de problème de conception.

Après avoir montré l'intérêt des techniques de programmation par contraintes pour la conception, nous examinerons les conditions nécessaires pour la résolution d'un environnement de conception coopérative à base de contraintes puis nous détaillerons le cœur de notre système constitué d'un algorithme de résolution de conflits. Nous proposons enfin un exemple d'application par le biais d'un prototype de système de conception collaborative d'une transmission de puissance par engrenages.

2. RESOLUTION DE PROBLEMES DE CONCEPTION A L'AIDE DE CSP

2.1. Les CSPs

Un CSP est défini par un triplet (V, D, C) tel que:

– $V = \{v_1, v_2, v_3, ..., v_n\}$ est un ensemble fini de variables dites variables contraintes. n étant le nombre entier de variables du problème à résoudre.

– $D = \{d_1, d_2, d_3, ..., d_n\}$ est un ensemble fini de domaines de valeurs des variables de V tel que pour chaque v_i de V, v_i prend ses valeurs dans d_i.

– $C = \{c_1, c_2, c_3, ..., c_p\}$ est un ensemble fini de contraintes. p étant un entier quelconque, représentant le nombre de contraintes du problème. Chaque contrainte $c_i(v_{i1}, v_{i2}, ..., v_{ik})$ de C porte sur un sous ensemble $\{v_{i1}, v_{i2}, ..., v_{ik}\}$ de V.

Avec i et k entiers finis tels que : $0 < i < p+1$ et $0 < k < n+1$.

Résoudre un CSP revient à instancier chacune des variables de V tout en satisfaisant l'ensemble C des contraintes du problème.

On trouve dans la littérature plusieurs types de CSP :

- Les CSP discrets : Les méthodes CSP sont des méthodes issues de la recherche opérationnelle et l'intelligence artificielle. Les premiers travaux datent de plus de trente ans. Citons (Waltz, 1972), (Montanari, 1974), (Laurière, 1976) et (Macworth, 1977). Ces méthodes CSP discrètes, de complexité exponentielle, sont basées sur l'énumération et le filtrage. Ce filtrage, appelé aussi propagation des contraintes, permet de réduire les domaines de définition des variables au fur et à mesure de la résolution. Mais il faut faire un compromis entre ce prétraitement et l'énumération. On se contente donc de faire une propagation locale, en général la cohérence d'arc : à chaque nuplet de variables en relation, on supprime les instances qui ne respectent pas cette contrainte.

Les techniques de résolution propres à la technologie informatique des CSP sont basées sur la propagation de contraintes. Associons à chaque variable du problème un axe et constituons ainsi un hyperespace de dimension n. L'algorithmique de propagation de contraintes permet de donner en permanence l'hyper parallélépipède rectangle dans lequel est inscrit le domaine des solutions du problème. Ceci revient à projeter sur les différents axes de l'hyperespace des variables du problème le domaine des solutions et à retourner les intervalles de valeurs ainsi générés.

La communauté CSP a développé des axes de recherche utiles au concepteur (Mulyanto, 2002). Les CSP dynamiques permettent d'ajouter ou de retrancher une ou des contraintes. Cela permet de traiter des problèmes de configuration pour la gestion des options d'un produit industriel (comme par exemple dans l'industrie automobile) comme dans (Aldanondo et al, 2003). Les CSP dynamiques permettent également de résoudre des problèmes mixtes de dimensionnement et de configuration comme dans (Yvars , 2001).

- Les CSP continus : Pour dépasser la limitation de ces CSP à variables discrètes, on a développé des CSP avec des domaines à variables continues à valeur dans des intervalles réels. La propagation des contraintes se fait alors sur les intervalles puis il y a dichotomie. Cette technique de résolution par intervalles est une synthèse entre l'analyse par intervalles de (Moore, 1966) et les CSP dans (Davis, 1987), (Falting, 1994), (Lhomme, 1993), (Lhomme et Rueher , 1997). Plusieurs techniques ont été mises au point et on en trouvera une présentation dans (Delobel, 2000).

2.2. Adéquation des CSPs pour la résolution de problèmes de conception

De manière générale, un problème de dimensionnement en conception consiste à déterminer l'ensemble des valeurs des variables de définition du produit une fois les options choisies. Il s'agit donc le plus souvent en mécanique d'un problème de satisfaction

de contraintes en variables mixtes (variables discrètes et variables continues) avec des contraintes variées (linéaires, quadratiques, non linéaires …) (Vareilles, 2005), (Yvars, 2005).

Les problèmes de configuration de produits quant à eux consistent à choisir les différentes options de configuration d'un produit en fonction d'un cahier des charges client (par exemple pour une gamme d'automobile donnée, la couleur, le type d'autoradio, le nombre de portes…) en tenant compte des éventuelles incompatibilités et/ou interdépendances entre options. Il s'agit donc souvent d'un problème de satisfaction de contraintes à variables discrètes voire même booléennes avec des contraintes logiques (Hadj Hamou, 2002) (Aldanondo, et al, 2003).

3. SPECIFICATION POUR UN ENVIRONNEMENT DE CONCEPTION COLLABORATIVE A BASE DE CONTRAINTES

3.1. Cadre des travaux

Nous cherchons à mettre au point des modèles algorithmes et outils informatiques susceptibles de traiter le problème suivant :

Une équipe de concepteurs travaille de manière collaborative à l'instanciation d'un même artefact.
Le système devra être capable de capter en temps réel les décisions des concepteurs concernant les variables du produit, de les propager et de gérer les incompatibilités entre décisions de manière à converger vers un artefact totalement instancié en minimisant les décisions remises en cause (cf figure 1).

Dans ce cadre, nous distinguons deux types d'approche :

Résolution supervisée: dans laquelle, le savoir-faire de conception serait utilisé comme connaissance de supervision aidant à détecter à priori les problèmes en surveillant les décisions prises par les concepteurs.

Résolution non supervisée: dans laquelle aucun contrôle de supervision n'est implémenté et où chaque concepteur peut avoir directement une action sur le produit en cours de conception.

Plusieurs travaux ont été réalisés en matière d'ingénierie collaborative, essentiellement sur les systèmes de gestion de base de données qui ont donné naissance aux concepts actuels de PDM et PLM (Grieves, 2005).

D'autres se sont concentrés sur la mise au point syntaxique et sémantique des modèles de conception collaboratifs de manière à éviter au plus tôt les conflits entre concepteurs dès l'étape de modélisation notamment à base d'ontologies (Lima et Ghodous, 2007).

Malheureusement, il existe à l'heure actuelle très peu de systèmes de gestion temps réel de conflits de décisions de conception. Quant aux systèmes de conception collaborative à base de CSP ils sont encore plus rares. A notre connaissance, ils se résument aux travaux concernant les systèmes de contraintes distribués (Telerman et al, 2005)

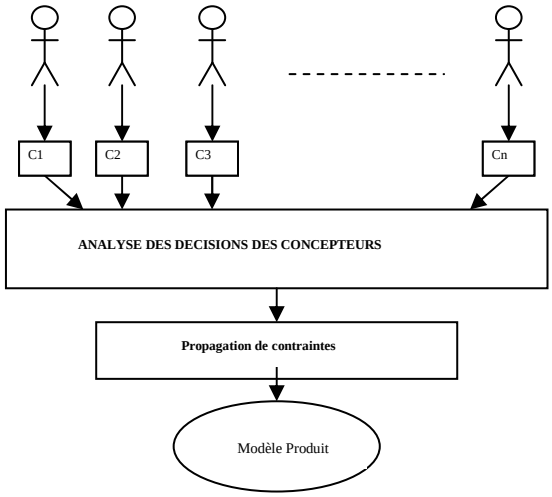

Figure 1. Architecture générale

4. UN ALGORITHME POUR LA GESTION DE DECISIONS DE CONCEPTION

4.1. Objectifs de l'algorithme

Le rôle de l'algorithme est tout d'abord de prendre en compte toutes les décisions prises par les différents concepteurs à un instant donné. Chaque décision étant une contrainte sur le produit, des incompatibilités entre décisions risquent rapidement d'apparaître. C'est pourquoi, après avoir pris en compte toutes les décisions prises à un instant donné, l'algorithme doit ensuite combiner ces différentes décisions entres elles et les tester (par propagation de contrainte sur le modèle du produit) pour enfin retourner la plus longue combinaison de décisions pouvant être appliquées au produit.

En terme de contraintes, le problème est en analogie avec la détermination d'une K-cohérence des décisions de conception en maximisant K.

Pour remplir son rôle, l'algorithme doit donc générer les différentes combinaisons à appliquer entres les décisions. De l'efficacité de cet algorithme dépend la durée de traitement globale.

Chacune des combinaisons sera soumise au réducteur de domaine (CSP) qui retournera un succès ou un échec. En

cas d'échec, la combinaison ne sera pas retenue. En cas de succès elle sera acceptée. L'efficacité de l'algorithme est entre autre basée sur les excellentes performances temps réel des réducteurs de domaines actuels.

Les hypothèses retenues pour la résolution sont les suivantes :

- Une décision acceptée ne sera pas remise en question par la suite.

- Chaque concepteur ne prend qu'une seule décision à la fois.

On trouve ci-dessous l'algorithme de gestion des décisions en pseudo langage :

```
Début
Pour (D Є ListeDecision)
    ListeNumero[i]_D.GetNumero()
Fin Pour
Si (TestEnsemble(ListeDecision))
Alors
    CombinaisonAcceptee_ListeNumero
Sinon
    CombinaisonRefusee_ListeNumero
    TestCombinaison(ListeDecision)
Fin Si
LaCombinaisonRetenue_
ChoixCombinaisonAcceptee ()
Si (LaCombinaisonRetenue = vide)
Alors
    Pour (D Є ListeDecision)
    ListeDesDecisionRefusee.Ajout(D)
    Fin Pour
Sinon
    Pour (D Є ListeDecision)
    Fait_0
    Si (D.GetNumero()ЄLaCombinaisonRetenue)
    Model.AjoutContrainte(D)
    Fait_1
    Fin Si
    Si (Fait=0)
    ListeDesDecisionRefusee.Ajout(D)
    Fin Si
    Fin Pour
Fin Si
Fin
```

4.2. Fonctionnement de l'algorithme

Pour une génération aisée des différentes combinaisons, on a décidé d'utiliser un algorithme qui va générer des combinaisons de nombres entiers (de 1, 2, 3 jusqu'au nombre de décisions). Les décisions étant au préalable numérotées, pour les combiner selon la combinaison d'entiers générée il suffit alors de comparer le numéro porté par les décisions et ceux présents dans la combinaison générée.

Le choix a été fait de générer les combinaisons avec un nombre d'éléments croissant. Ainsi, on va tester dans un premier temps chaque décision indépendamment, puis on va tester les combinaisons de deux décisions et ainsi de suite jusqu'à trouver la plus longue combinaison de décisions pouvant être satisfaites.

La génération des combinaisons pour tester les décisions se fait de manière systématique, sans utiliser d'heuristiques particulières concernant par exemple la cohérence des décisions ou encore prenant en compte des règles métiers. Cet aspect pourra faire l'objet de développements futurs.

Pour que l'algorithme de gestion des décisions soit efficace, il ne faut pas générer des combinaisons qui ont déjà été générées précédemment mais il ne faut pas non plus générer toutes les combinaisons possibles.

De plus, il faut tenir compte du fait qu'une combinaison dont les éléments sont en fait la permutation d'une autre ne sera pas pertinente car un réducteur de domaine ne tient pas compte de l'ordre d'apparition des décisions. On pourra donc, si possible, éviter de la générer pour ne pas perdre de temps à la tester.

De plus au fur et à mesure des différents tests de combinaisons entre les décisions, certaines décisions ou combinaisons de décisions peuvent se révéler incompatibles avec le produit. On pourra donc tenir compte de cela pour les tests futurs et gagner du temps en évitant les cas d'échec. Tester une combinaison dont on sait à l'avance qu'au moins une des décisions ou une combinaison ne pourra pas être satisfaite est inutile.

Etant donné, qu'il s'agit donc d'une part, de générer des combinaisons, et d'autre part, de les tester tout en prenant en compte les cas d'échec, il a été décidé de suivre cette logique et de développer un algorithme qui va générer les combinaisons puis d'effectuer un test préalable de la combinaison en appliquant une fonction de filtrage qui aura mémorisé les cas d'échec, avant de soumettre au réducteur de domaine (propagation de contrainte) la combinaison de décisions correspondante. En cas de détection d'un cas d'échec par le filtre, une nouvelle combinaison sera générée sans que soit sollicité le réducteur de domaine.

L'intérêt d'utiliser ainsi une telle fonction de filtrage située après la génération de la combinaison et avant la sollicitation du réducteur au lieu de tenir compte des cas d'échec directement dans l'algorithme de génération des combinaisons, est d'alléger la génération de combinaisons. En fait, il a été décidé d'exploiter les fonctions de la programmation par contraintes pour générer les différentes combinaisons. La génération des combinaisons utilise donc son propre réducteur de

domaine et tenir compte des cas d'échec à ce niveau alourdirait l'algorithme ainsi que la génération.

L'application du filtre se résume alors à effectuer une recherche dans une liste de cas d'échec (liste d'entiers ou de combinaisons d'entiers) pour vérifier si des éléments de la combinaison générée n'y figurent pas. Cette liste sera bien sûr mise à jour au fur et à mesure de la découverte d'autres cas d'échec lors des différents tests effectués par propagation de contraintes.

5. APPLICATION A LA CONCEPTION COLLABORATIVE D'UN TRAIN D'ENGRENAGES

5.1. Problématique

Un train d'engrenage est un ensemble d'arbres composés d'engrenages permettant de transmettre une puissance. Un tel composant sera défini pour le cahier des charges ci dessous :

- La puissance à transmettre.
- La vitesse de rotation en entrée du train.
- La vitesse de rotation en sortie du train.
- Les sens de rotation en entrée et en sortie de train (identiques ou inverses).

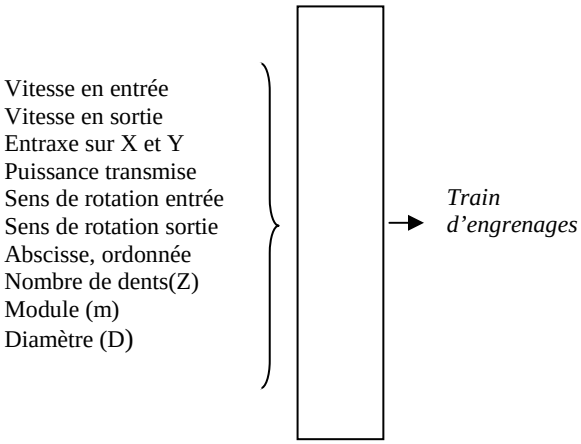

Figure2. Dimensionnement d'un train d'engrenages

Le dimensionnement terminé (cf. figure 2), nous souhaitons disposer des :

- Nombre d'arbres du train.
- Implantations des arbres sous la forme de coordonnées.
- Caractéristiques de chaque roue (Nombre de dents, Module et diamètre).

Des contraintes d'encombrement pourront être précisées et un critère de coût optimisé (dans notre cas, nombre d'arbres et valeurs minimales des caractéristiques des roues).

Du point de vue des CSP, il s'agit d'un problème mixte à variables discrètes et continues.

5.2. Modélisation du produit

Un train d'engrenages peut être vu comme un composant, composé d'un ensemble d'arbres. Chacun des arbres étant lui même composé de roues.

Une analyse orientée objet élémentaire montre que trois classes sont nécessaires pour représenter un train d'engrenages. Chaque classe encapsule un ensemble de variables contraintes sous forme d'attributs ainsi qu'un jeu de contraintes. De plus les attributs sont mixtes : certains sont entiers d'autres flottants (réels).

Les contraintes du problème ont été volontairement limitées dans cet exemple aux relations nécessaires au pré dimensionnement d'une transmission de puissance par engrenages.

Nous obtenons les classes suivantes :

- La classe Trans : une instance de cette classe représente un train d'engrenages complet. Chaque « Trans » est composée d'un ensemble d'instances de la classe Arbre.
  Les contraintes globales à la transmission seront spécifiées à ce niveau : sens de rotation, encombrement
  Les contraintes inter arbres seront posées au niveau de cette classe :
  - Positionnement des arbres
  - Rapports de vitesse entre roues
  - Identité des modules entre roue menante et roue menée

- La classe Arbre : chaque Arbre est constitué d'une ou deux instances de la classe Roue

- La classe Roue : Chaque Roue est caractérisée par un nombre de dents (Z), un diamètre (D) et un module (M). Toute instance de la classe Roue est assujettie au respect des relations (pour un engrenage sans déport de denture) :
  - $D = M * Z$
  - $M \geq 1.45 * \sqrt{\frac{Ft}{k * \sigma f}}$

  contrainte de durée de vie de la roue entre le module, l'effort tangentiel, le

facteur de fiabilité et la Valeur de la contrainte en fonctionnement en pied de dent.

Ces relations sont considérées en tant que contraintes de dimensionnement.

Afin de concevoir un produit (ici un train d'engrenages) et lui imposer des contraintes, il faut au préalable définir les caractéristiques générales propres au produit. Il s'agit de fixer les paramètres permettant d'instancier le produit et de définir son domaine d'étude en fixant les bornes ou les valeurs pouvant être prises par les variables contraintes du produit.

Pour le train d'engrenages, les paramètres permettant d'instancier le produit sont :

- la vitesse Ne de l'arbre d'entrée,
- la vitesse Ns de l'arbre de sortie,
- le nombre Nb d'arbres (sinon il faut préciser le sens de rotation de l'arbre de sortie par rapport à l'arbre d'entrée).

Les paramètres permettant de définir le domaine d'étude du train d'engrenages sont :

- Les valeurs minimale et maximale pour les vitesses de rotation des arbres.
- Les valeurs minimale et maximale pour les diamètres des roues.
- Les valeurs minimale et maximale pour le nombre de dents des roues.
- Les valeurs minimale et maximale pour les largeurs des roues.
- Les valeurs minimale et maximale pour les angles entre les centres des arbres.
- Les valeurs minimale et maximale pour les angles d'hélice des roues.
- Les valeurs pouvant être prises par les modules.
- La valeur maximale pour la distance entre les roues d'un même arbre.

5.3. Instanciation collaborative du produit

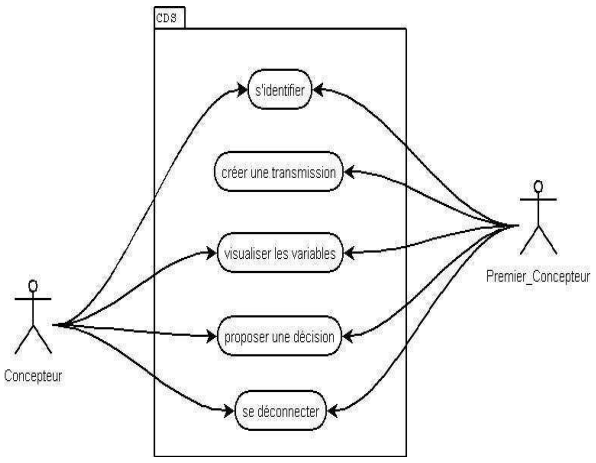

Figure 3. Use case du prototype

Notre prototype a été implémenté en C++ sur la base de la librairie de programmation objet par contraintes IlogSolver (Ilog, 2006). Un serveur intranet a été mis en place et permet la conception simultanée d'un train d'engrenages par l'intermédiaire d'un simple navigateur (Dameron et al, 2007). Les données sont gérées en php via le système de gestion de base de données MySql.

Les différentes fonctionnalités du prototype sont illustrées par le diagramme UML de cas d'utilisation de la figure 3.

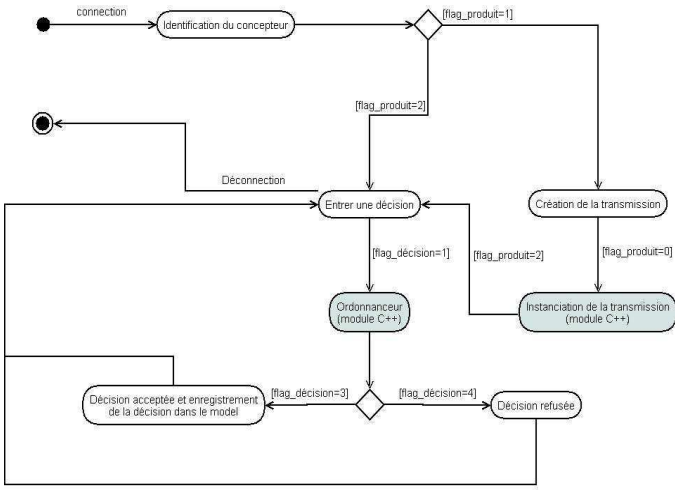

Figure 4. Diagramme d'activité du prototype

Le diagramme d'activité UML de la figure 4 illustre les séquences suivies par un concepteur utilisateur du prototype sur un intranet via un navigateur banalisé.

En terme d'interface utilisateur, la première page de l'interface permet au concepteur de s'identifier (en entrant son nom) ce qui va l'ajouter a la base de données ou bien l'instancier de nouveau si il s'y trouvait déjà.

Dans le cadre du prototype, l'utilisateur n'a pas pour le moment à rentrer de mot de passe. La gestion de sessions et d'identification d'utilisateur en php est quelque chose de bien connu et de maîtrisé. Nous avons choisi de nous concentrer sur l'essentiel pour ce démonstrateur.

Dans un second temps, le programme d'interface vérifie certains points :

· Si le concepteur qui se connecte, est le premier, il est redirigé vers une page lui permettant de créer la

transmission puis vers une page d'attente jusqu'à l'instanciation du produit par le noyau de résolution écrit en C++. Ensuite, il est dirigé vers la page des variables du produit.

· Si le concepteur n'est pas le premier mais que le produit n'est pas encore instancié, il est redirigé sur une page d'attente jusqu'à l'instanciation du produit puis il est dirigé vers la page des variables du produit.

· Si le produit existe déjà, le concepteur est directement dirigé vers la page des variables du produit.

Une fois sur la page des variables du produit, l'interface présente l'ensemble des variables présentes dans la base de données en affichant :

· Si la valeur a été contrainte par un cycle de calcul, l'interface affiche sa valeur en vert ainsi que le nom du concepteur ayant pris la décision. Cette valeur ne peut pas être modifiée.

· Si la valeur n'a pas été contrainte, la valeur proposée par le résolveur de CSP est affichée a titre d'information en noir. Ainsi, le concepteur peut poser une contrainte sur cette variable (réduction de son domaine de valeur ou instanciation de la variable).

Une fois la contrainte posée, l'interface se met en phase d'attente (Dans le cadre de ce premier démonstrateur, le concepteur ne peut prendre les décisions que une par une). Le calcul du réducteur de domaine effectué, le concepteur est informé de l'état de sa décision et la mise à jour de la liste des variables est effectuée en y incluant les nouvelles décisions acceptées.

Chaque concepteur peut alors revenir à la liste des variables afin de poser une nouvelle contrainte.

Enfin, un bouton de déconnection permet à chacun des concepteurs de se déconnecter de l'interface.

6. CONCLUSION

Nous avons présenté les principes de fonctionnement d'un prototype de logiciel d'aide à la conception collaborative sous contrainte non supervisée de produit industriel. Un prototype a été réalisé pour illustrer pratiquement ces principes sur le cas d'un dimensionnement de transmission de puissance par engrenages.

Ce papier jette les bases de la généralisation de l'approche CSP, déjà validé en mono conception, au cas d'une conception multi acteurs.

Les évolutions des travaux portent actuellement sur la généralisation de l'algorithme de gestion des décisions de conception en terme de stratégies disponibles ainsi que sur l'intégration d'un méta modèle produit sous contraintes. Ce dernier sera basé sur les travaux réalisés autour du projet KoMod (Yvars, 2006),

Enfin, les perspectives de ces travaux s'orientent également sur l'intégration de savoir-faire de résolution en multi conception de manière à disposer d'un système de conception supervisé par les connaissances, avec l'espoir d'améliorer les performances dans la gestion des décisions de conception.

REFERENCES

Aldanondo M., Hadj-Hamou K. et Lamothe J., 2003. Product generic modelling for configuration : Requirement analysis and modelling elements, *KOGAN PAGE*, p.50-70,ISBN 1-9039-9643-0

Barricelli N. and all ,1954. Esempi numerici di processi di evoluzione. *Methodos*: 45-68.

Bernard A.,2000. Modèles et approches pour la conception et la production intégrées, *Revue APII - JESA, Vol. 34 - 2-3/2000, pp 163-194, N°ISBN : 2-7462-01445.*

Chandrasekaran, B., 1990. Design Problem : a task analysis, *AI Magazine*, Vol. : Winter.

Clerc M. ,2005. L'optimisation par essaims particulaires. Versions paramétriques et adaptatives, *Hermés Science*.

Cvijovic D., Klinowski J. ,1995. Taboo search - an approach to the multiple minima problem. *Science 267*, 664-666

Dameron V., Foray, Minet, Nguyen et Soret, 2007. Logiciel d'aide à la conception collaborative de trains d'engrenages, *Mémoire de fin d'études SupMeca*.

Davis E.,1987. Constraint propagation with interval labels, *Artificial Intelligence*, v. 24.3.

Delobel F.,2000. Résolution de systèmes de contraintes réelles non linéaires, *Phd Thesis Computer Science*, Université de Nice Sophia Antipolis, France.

Dorigo M.,1992. 'Optimization, Learning and Natural Algorithms', *PhD thesis*, Politecnico di Milano, Italy.

Dorigo M. & T. Stützle, 2004. Ant Colony Optimization, *MIT Press*. ISBN 0-262-04219-3

Falkenauer E. ,1997. Genetic Algorithms and Grouping Problems, *Chichester, England: John Wiley & Sons* Ltd. ISBN 978-0-471-97150-4

Falting B. ,1994. Arc consistency for continuous variables, *Artificial Intelligence* 65(2).

Grieves, 2005, Product LifeCycle Management, *Mc Graw Hill Edition*, ISBN 978-0-07-145230-4.

Hadj-Hamou K., 2002, Contribution à la conception de produit à forte diversité et de leur chaîne logistique : une approche par contraintes, *Thèse de Doctorat d'Université* INPT, Toulouse.

Ilog ,2006. IlogCP, Reference Manual,*Ilog*, Gentilly, France.

Kirkpatrick S.,Gelatt C.D. and Vecchi M.P.,1983. Optimization by Simulated Annealing, *Science*, Vol 220, Number 4598, pages 671-680.

Kota S., Ward A.C. ,1991. Functions, structures and constraints in conceptual design,Design Laboratory, department of Mechanical Engineering and applied Mechanics University of Michigan.

Lauriere J.L.,1976. Un langage et un programme pour résoudre et énoncer des problèmes combinatoires: ALICE, *Phd Thesis*, Paris 6 University, France.

Lhomme O.,1993. Consistency Techniques for Numeric CSPs, *13th International Conference on Artificial Intelligence*, pp 232-238, Chambéry, France.

Lima dutra M., Ghodous P.,2007. A reasoning Approach for conflict dealing in collaborative design, 14th ISPE Conference on concurrent engineering, Sao José dos Campos, Brésil.

Lhomme O.-M., Rueher ,1997. Application des techniques CSP au raisonnement sur les intervalles, *Revue d'intelligence artificielle*, Dunod, Vol. 11:3, pp. 283-311.

Mackworth A.K.,1997. Consistency in networks of relations, *Artificial Intelligence* 8, 1, 1977, pp.99-118.

Montanari U.,1974. Networks of constraints: fundamental properties and applications to pictureprocessing, *Information Science* 7, 1974, pp.95-132.

Moore R.E. ,1966. Interval Analysis, *Prentice-Hall*.

Mulyanto T., 2002, Utilisation des techniques de programmation par contraintes pour la conception d'avions, *Thèse de Doctorat* ENSAE, Toulouse.

Sveda M.,2007. Industrial Measurement Application Development Using Case-Based Reasoning, *Proc of Second International Conference on System (ICONS'07*), IEEE Computer Society.

Telerman V., Preis S., Snytnikov N., Ushakov D., 2005, Collaboration Model Based on Interval Constraints, *Proceedings of 1st International Workshop on Distributed ans Speculative Constraint Processing, DSCP'05.*

Vareilles E., 2005, Conception et approches par propagation de contraintes : Contribution à la mise en œuvre d'un outil d'aide interactif, *Thèse de Doctorat* INPT, Toulouse.

Waltz D. ,1972. Generating semantic descriptions from drawings of scenes with shadows, *MIT*, Massachusetts.

Yvars P.A. ,2001. Contribution à la représentation des connaissances en ingénierie intégrée de produits et de systèmes de production, *Mémoire d'habilitation à diriger les recherches, INPG*, Grenoble, january.

Yvars P.A.,2005. Eléments pour une ingénierie supervisée par les contraintes, *Proceedings Congrés Français de Mécanique CFM'05*, UTT, Troyes, France.

Yvars P.A.,2006. A three level architecture for kwnoledge representation in mechanical engineering, *Proceedings of 6th International Conference on Integrated Design and Manufacturing in Mechanical Engineering, IDMME'06*, Grenoble, France.