# L'utilisation des techniques de gestion de production dans la construction d'un grand detecteur de physique

Steven Murray

Laboratoire d'Annecy-le-Vieux de Physique des Particules

Thèse
présentée à
l'Université de Savoie
pour obtenir le titre de
Docteur de l'Université de Savoie
Spécialité: instrumentation
par

**Steven MURRAY**

# L'utilisation des techniques de gestion de production dans la construction d'un grand détecteur de physique

# The use of production management techniques in the construction of a large high energy physics detector

Soutenue le **10 novembre 2000** devant le jury composé de

| | | |
|---|---|---|
| Mr D. Décamp | | Professeur à l'Université de Savoie |
| Mr F. Dupont | | Docteur ès-Sciences |
| Mr L. Foulloy | | Professeur à l'Université de Savoie |
| Mr J-M. Le Goff | | Professeur à University of the West of England |
| Mr D. Linglin | Rapporteur | Directeur de Recherches |
| Mr R. McClatchey | Rapporteur | Professeur à University of the West of England |
| Mr G. Organtini | | Docteur ès-Sciences |
| Mr J-P. Vialle | Directeur de Thèse | Directeur de Recherches |

# Résumé

Le cycle de vie d'un grand détecteur pour la physique des Hautes Energies peut être divisé en quatre phases: la conception, le dessin, la construction, et le fonctionnement/maintenance. Des systèmes informatiques existants permettent de gérer la conception et le dessin détaillé, et d'autres l'enregistrement des données du détecteur et leur analyse. Cependant, l'équivalent n'existe pas pour la phase de construction, où seules des solutions particulières à chaque problème ont été utilisées.

Les détecteurs de la prochaine génération à construire pour les expériences auprès du futur accélérateur LHC (Large Hadron Collider) du CERN (Laboratoire Européen de Physique des Particules) à Genève seront dix fois plus complexes que les détecteurs existants, et leur cycle de vie s'étendra sur une vingtaine d'années. Cette complexité croissante associée une durée de vie plus grande impose des solutions informatiques solides et évolutives pour maîtriser le processus de construction.

En conséquence, dans cette thèse nous avons étudié et appliqué les techniques de contrôle de production à la phase de construction d'un grand détecteur pour la physique. Cette thèse et le travail associé sont une contribution à la conception et à la mise en œuvre du projet CRISTAL (Coopérative Repositories and Information System for Tracking Assembly Lifecycle, soit en français: Système d'Information et de Bases de données fédérés pour le suivi de construction), le premier logiciel qui ait été conçu pour saisir les caractéristiques et enregistrer la totalité de l'histoire individuelle de chaque composant d'un détecteur pendant la phase de construction.

Dans sa conclusion, cette thèse montre comment les techniques développées peuvent être appliquées au monde de l'industrie.

# Remerciements

Ces travaux de thèse ont été menés à bien au Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP) pendant les années 1997 à 2000. Je voudrais remercier tout le service informatique ainsi que les personnes du lapp qui m'ont aidé pendant ce temps. Mes remerciements vont également à mes collègues du centre européen pour la recherche nucléaire (CERN) qui ont travaillé avec moi sur le projet CRISTAL. Il m'est impossible d'énumérer tout le monde, toutefois je voudrais exprimer ma plus grande gratitude aux amis et aux collègues suivants:

Mon Directeur de Thèse, le Dr. Jean-Pierre Vialle pour m'avoir donné l'opportunité de mener à bien mes travaux de thèse au LAPP. Ce travail a été effectué dans le contexte du projet CRISTAL - une collaboration entre le LAPP et le centre européen pour la recherche nucléaire (CERN). Jean-Pierre était l'un des fondateurs principaux de cette collaboration.

Le Dr. Jean-Marie Le Goff pour son action incitative en vue de la création du projet de CRISTAL au CERN.

Alain Bazan et Thierry Le Flour pour leur amitié, connaissances et conseils, sans qui cette thèse n'aurait pas été possible.

Le Dr. Richard McClatchey pour son aide incommensurable dans la rédaction de ce texte. J'ai profité considérablement de son expérience et de ses conseils pendant ce travail de recherche pour la réalisation de cette thèse et des documents que j'ai rédigés.

Le Dr. Zsolt Kovacs pour sa compréhension profonde du projet CRISTAL, et son enthousiasme sans fin pour les nouvelles technologies qui ont été bénéfiques à mon travail et m'ont apporté une grande expérience.

Sophie Lieunard pour avoir partagé son bureau avec moi et qui m'a donc accepté pendant ces 3 dernières années.

Florida Estrella, Peter Brooks, Christophe Koch, Guy Chevenier, Márton Zsenei, Giovanni Organtini et László Vargo pour leur travail de développement du logiciel CRISTAL.

Myriam Froger, ma "mère au LAPP", pour son aide dans les démarches administratives.

En conclusion, et en tout premier, Sylvia et ma mère pour m'avoir supporté pendant des heures non-travaillées, me donnant la force et la motivation requises pour terminer ce travail.

# Table des matières

# Liste des figures

# Liste des tables

# 1 Introduction

## *1.1 Préface*

La physique expérimentale des Hautes Energies (HEP) emploie de grands accélérateurs et détecteurs de particules pour tester les théories qu'elle développe sur la structure de la matière. Comme ces tests exigent des énergies de plus en plus élevées, la taille, la complexité et la durée des accélérateurs de particules et des détecteurs augmentent. Avec l'arrivée du nouveau grand projet de Collisionneur de Hadrons (LHC) au CERN en Suisse, les détecteurs contiendront dix fois plus de composants que ceux construits précédemment et auront des durées de fonctionnement d'environ 15-20 ans. Le cycle de vie d'un grand détecteur de particules, tel que celui de l'expérience Compact Muon Solenoid (CMS) actuellement en construction au CERN, peut être divisé en trois phases:

1. Conception & Plans
2. Construction
3. Prise de données & Maintenance

Des systèmes informatiques généraux ont été développés pour suivre le cycle de vie du détecteur. Cependant ceux concernés par les phases de conception et de construction se sont concentrés uniquement sur le contrôle des documents qui décrivent le détecteur. Ceci a mené à une vue du détecteur basée sur les types de composants qui le constituent. Il n'y a eu aucun système général développé qui puisse assurer le suivi des différents composants physiques. En particulier il n'y a aucun système établi pour enregistrer les caractéristiques et l'histoire de l'assemblage de chaque composant individuel du détecteur, condition qui doit être satisfaite pour accomplir l'étalonnage précis du détecteur et son contrôle de manière efficace. Historiquement le CERN a créé les solutions ad-hoc, sans doute parce que chaque détecteur est unique, et que produire un système logiciel d'usage universel exige une analyse fouillée des principes de la construction d'un détecteur, une chose impensable avec les échelles de temps de la précédente génération du détecteur. Cette approche ad-hoc marchait dans le passé grâce à la taille limitée des détecteurs, mais la complexité encore plus grande et la longue durée de vie de la prochaine génération du détecteur exigera une solution plus générale et évolutive. En outre, la construction elle-même, impliquant des milliers de composants et durant plusieurs années, doit être gérée d'une façon plus automatique que dans le passé, pour guider efficacement les acteurs pendant cette phase. Cette thèse étudie donc l'application des techniques de gestion de production pour la construction d'un grand détecteur HEP.

## *1.2 Cadre de Recherche*

Cette thèse a été effectuée dans une équipe de recherche du Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP) de 1997 à 2000, dans une collaboration scientifique créée pour produire un système d'assistance logicielle pour la construction du détecteur électromagnétique de l'expérience CMS (ECAL). Le logiciel est nommé CRISTAL, ce qui signifie "Système d'information et d'enregistrement en coopération pour le suivi du cycle de construction". CRISTAL est le premier système logiciel général qui a été spécifiquement conçu pour suivre la

construction d'un détecteur au niveau de composant physique individuel du détecteur [Bazan et al, 1998]. ECAL est un sous-détecteur de l'expérience CMS au CERN[1]. Les membres de la collaboration sont:

- Conseil Européen pour la Recherche Nucléaire (CERN)
- Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP)
- University of the West of England (UWE)

Avec la contribution du MTA-KFKI à Budapest et de l' INFN à Rome.

## *1.3 Problème de Recherche*

La construction de grands détecteurs de physique se fait sur une longue période, ne se fait qu'une fois, et les tâches sont réparties à l'échelle mondiale. ECAL, qui sera construit à partir de 1999 jusqu'en 2005, est l'un de ceux là. Il sera construit en collaboration entre la France, l'Italie, la Russie et la Suisse. Un des aspects importants de la construction d'un grand détecteur de physique est que les caractéristiques et l'historique d'assemblage de chaque composant individuel du détecteur doivent être saisies et enregistrées de manière sûre. Les données doivent être recueillies au bon moment dans le processus de construction et être enregistrées au bon endroit, car certaines mesures ne peuvent plus être faites ultérieurement. La gestion et le stockage des données doivent faire face à la grande quantité de données (approximativement 1 Terabyte pour ECAL) et au fait que les données devront pouvoir être stockées et/ou extraites pendant une longue période, c'est à dire pendant la construction du détecteur, son exploitation et son entretien.
C'est la première fois qu'un système logiciel général est développé pour aider à la construction du détecteur au niveau des composants physiques individuels du détecteur. Par conséquent les techniques de développement requises ne sont pas encore toutes entièrement définies. Cette thèse se propose d'étudier et de développer ces techniques. La thèse a trois objectifs. Elle étudiera quels sont les besoins d'assistance, et elle étudiera et développera les moyens techniques appropriés à:

1. **Modéliser la description des composants du détecteur et le déroulement des opérations pour leur réalisation.**
2. **Visualiser le processus d'assemblage du détecteur et sa caractérisation.**
3. **Mesurer les capacités des ressources de fabrication dans un environnement de physique.**

En développant ces techniques cette thèse montrera que:
Une approche orientée vers la description devra être adoptée pour modéliser la composition d'un détecteur et le déroulement des opérations qui définissent et contrôlent son procédé de construction
Les Modèles de Conception sont des moyens efficaces pour développer des interfaces utilisateurs qui évoluent fréquemment.
Les techniques employées pour modéliser les applications industrielles peuvent s'appliquer à l'environnement de la physique pour créer un système souple capable de déterminer la capacité de production en temps réel.

La thèse conclut en montrant jusqu'à quel point ces techniques peuvent être utilisées dans d'autres systèmes logiciels pour la physique, et comment les technologies

[1] Voir l'appendice A pour une introduction aux accélérateurs, aux détecteurs, au CERN et à l'expérience CMS.

développées dans le projet de CRISTAL peuvent être transférées dans le domaine industriel.

## *1.4 Contribution personnelle de l'auteur dans ce travail de thèse*

Cette thèse a été conduite dans le cadre d'une équipe et par conséquent l'auteur n'a pas été seul à travailler sur les résultats présentés. Rapportons-nous aux trois objectifs de thèse de la section 1.3:

1. **Modélisation de la description des composants du détecteur et du déroulement des opérations pour leur réalisation.**
   L'auteur a collaboré à l'étude du modèle. L'auteur a, en particulier, effectué une étude approfondie dans le langage Unified Modelling Language (UML) incluant son architecture de métamodèles à trois couches.

2. **Visualisation du processus d'assemblage du détecteur et de sa caractérisation.**
   L'auteur a été le principal artisan et développeur du logiciel graphique d'interface utilisateur, logiciel servant aux techniciens construisant le détecteur dans les centres de production et d'assemblage.

3. **Mesure des capacités de production dans un environnement de physique.**
   L'auteur a fait tout le travail de recherche dans le domaine de la mesure de la capacité de production.

## *1.5 Méthodologie du projet*

La méthodologie utilisée pendant ce travail de thèse a été influencée par la méthode de travail de l'équipe CRISTAL. Cette section décrit donc la méthodologie employée par l'équipe CRISTAL. Elle discute les questions qui ont dû être abordées et présente les solutions choisies ainsi que les arguments pour leur sélection.

Les questions à résoudre par la méthodologie de projet choisie peuvent être divisées en deux catégories: celles concernant l'analyse et la conception du modèle, et celles concernant le cycle de développement du logiciel. Au départ, on savait qu'un système logiciel permettant la gestion de la production pendant la construction d'un grand détecteur de physique serait grand, complexe, devrait travailler de manière distribuée à l'échelle du monde, et que les échelles de temps seraient très grandes. Les phases d'analyse et de conception d'un tel projet de développement exigeraient une technique de modélisation pour faire face à de telles questions. Le fait que ce soit la première fois que des techniques de gestion de production industrielle seraient rigoureusement appliquées pendant la phase de construction d'un grand détecteur de physique, indiquait que les utilisateurs seraient dans l'incapacité de fixer l'ensemble des règles, des contraintes et des processus dès le début. Ceci a évidemment éliminé l'utilisation du modèle traditionnel de développement du logiciel dit « de la chute d'eau » [Sommerville, 1992], et donc une stratégie plus appropriée a dû être trouvée. Les exigences de l'utilisateur évoluent pendant la phase de construction car le détecteur est un objet unique. Les constructeurs acquièrent donc de l'expérience et ils améliorent la manière de construire le détecteur. En conséquence, des demandes de nouvelles fonctionnalités du système peuvent jaillir à tout moment de la construction. Le but

d'un physicien est de construire le meilleur détecteur dans le moins de temps et il ou elle ne veut pas être contraint par des décisions prises au début si les changements sont techniquement faisables.

La Technique de modélisation d'objet (OMT) de [Rumbaugh et al, 1991] a été choisie pour l'analyse et la conception du logiciel, par opposition aux techniques traditionnelles de décomposition fonctionnelle. OMT fournit, entre autres concepts orientés-objet, les idées d'encapsulation et d'héritage. L'encapsulation orientée-objet consiste à regrouper les données avec les méthodes qui les manipulent dans des entités appelées Objet. Ce type d'encapsulation fournit deux avantages par rapport à la décomposition fonctionnelle. Premièrement, les objets réels qui ont un état et sur lesquels on agit peuvent être modélisés plus naturellement et de manière plus proche, fournissant un modèle beaucoup plus intuitif. Deuxièmement, la modélisation d'un système distribué sous forme d'un ensemble d'objets distribués interconnectés construit un système de transmission puissant. Non seulement une méthode peut être appelée à un emplacement indiqué, mais les données sur lesquelles elle agit peuvent également être précisées. Dans le contexte du développement d'un système informatique pour suivre les différents composants d'un détecteur au long de la construction, il est naturel de modéliser une pièce sous forme d'objet parce qu'elle possède un état - par exemple l'historique de sa production et sa composition - et on agit dessus, par exemple par une mesure ou une action d'assemblage. Quel que soit le moment, il y aura un grand nombre de composants du détecteur dans un centre de production par rapport au nombre d'ordinateurs. Ceci veut dire que beaucoup de composants du détecteur auront leur représentation logicielle sur un même ordinateur. La communication d'objets distribués fournit donc la bonne méthode pour communiquer avec ces représentations. L'héritage orienté-objet fournit un mécanisme puissant pour la réutilisation de parties du logiciel, une nécessité de n'importe quel grand système logiciel complexe. Des objets contenant les mêmes structures de données et les mêmes méthodes sont spécifiés par une classe simple, où une classe peut être visualisée comme très similaire à un type de données abstrait. Le mécanisme orienté-objet d'héritage permet à une nouvelle classe d'hériter des structures de données et des méthodes de classe existante, donc de réutiliser le code de la classe existante

[Sorensen, 1995] montre qu'un modèle itératif de développement du logiciel permet d'obtenir les fonctionnalités désirées plus rapidement que le modèle traditionnel de la chute d'eau en effectuant l'analyse, la conception, et les phases de mise en place sur des sous-ensembles du projet global de développement logiciel. Une approche itérative du prototypage a été donc adoptée pour le cycle de développement du logiciel, afin de donner à des utilisateurs le support requis pour permettre l'évolution de leurs contraintes. L'idée est de fournir aux utilisateurs les fonctionnalités principales qu'ils ont demandées très tôt dans le processus de développement, de sorte qu'ils peuvent faire connaître leurs réactions et affiner leurs idées.

En conclusion, une approche de modélisation orientée-objet a été adoptée pour l'analyse et la conception du logiciel, et une stratégie de prototypage rapide itérative a été choisie pour le cycle de développement du logiciel.

## *1.6 Méthodologie de la thèse*

Une approche itérative a été adoptée pour affiner le sujet de cette thèse. Des utilisateurs ont été consultés de façon régulière, l'objectif de la thèse étant présenté, modifié et affiné à chaque fois. Entre ces consultations, l'auteur de la thèse étudiait le domaine actuel de recherches et les utilisateurs recevaient des informations sur ce qui avait été trouvé. Cette approche itérative a été adoptée pour s'assurer que le sujet et le travail de la thèse étaient utiles pour le projet et les utilisateurs de CRISTAL.

## *1.7 Structure de la thèse*

Le chapitre 2 analyse les contraintes de construction du calorimètre électromagnétique (ECAL). Le chapitre 3 présente l'état actuel des techniques de gestion de production utilisées au CERN et discute le besoin d'une assistance accrue pendant la phase de construction du détecteur. Les chapitres 4 à 8 présentent les techniques développées par cette thèse. Le chapitre 4 montre une architecture distribuée comme support à la construction du détecteur par une collaboration à l'échelle de la planète. Le chapitre 5 présente un modèle de données qui permet l'évolution des composants du détecteur et des processus de fabrication pendant la construction du détecteur. Le chapitre 6 montre comment développer une interface utilisateur graphique (GUI) complexe qui doit pouvoir évoluer fréquemment. Le chapitre 7 montre comment développer un GUI avec comportement dynamique complexe. Le chapitre 8 montre comment adapter les techniques industrielles de mesure de la capacité de production pour qu'elles puissent être utilisées pour la construction d'un détecteur HEP. Le chapitre 9 donne les conclusions de la thèse et présente des extensions possibles de ce travail dans le futur.

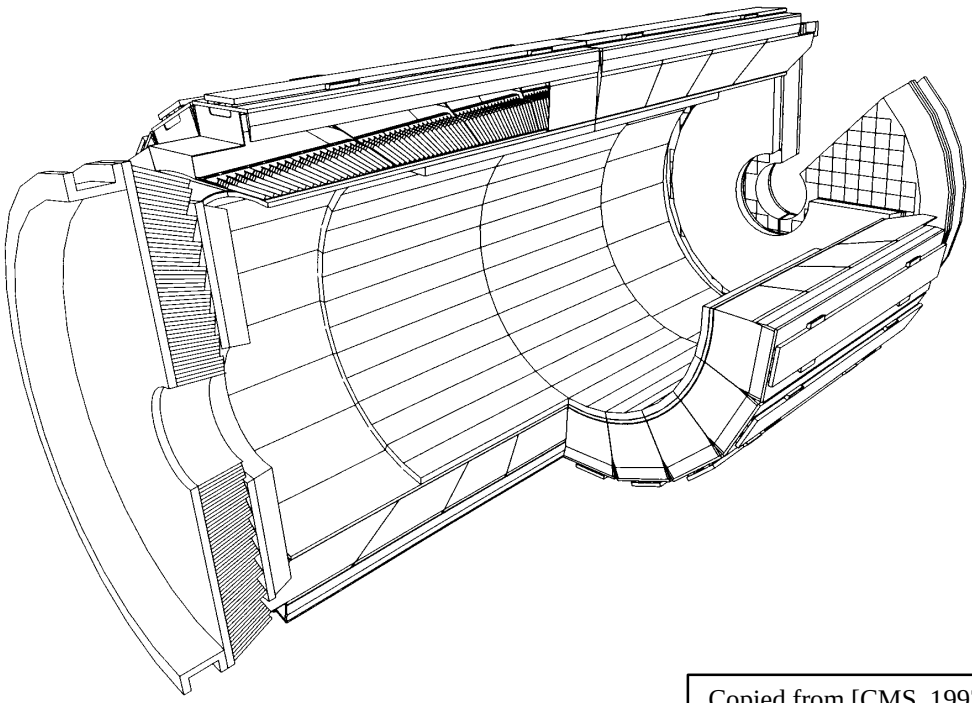
# 2 Analyse des spécificités liées à la construction du calorimètre ECAL

## *2.1 Introduction*

Ce chapitre analyse la construction du calorimètre électromagnétique (ECAL) pour identifier les conditions que doit remplir un système informatique d'aide à la construction. Le chapitre décrit la construction du ECAL en présentant la composition du détecteur, puis le processus actuel d'assemblage, et termine avec les particularités de la construction d'un grand détecteur de physique. La conclusion de ce chapitre énumère les conditions à remplir par un système d'aide à la construction, et montre qu'elles mettent en évidence le besoin d'une saisie de données contrôlée pendant la construction du détecteur, et la nécessité de mesurer la capacité de production.

## *2.2 Composition du ECAL*

Le ECAL sera formé de 61.200 cristaux de tungstate de plomb disposés dans un tonneau, comme présenté sur la figure 1. Au total il y aura environ un million de composants, soit un détecteur 10 fois plus complexe que la génération actuelle.



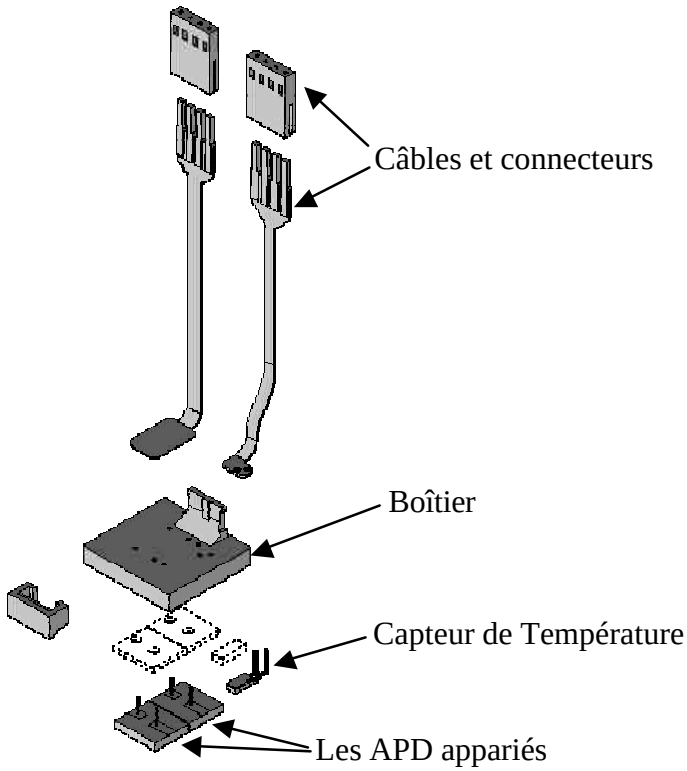
Copied from [CMS, 1997]

**Figure 1 Tonneau ECAL**

La réussite du ECAL dépend fortement de l'homogénéité de ses cristaux. Les processus de fabrication les plus évolués sont nécessaires pour produire ne serait-ce que quelques centimètres cubes de cristaux homogènes de tungstate de plomb. ECAL a besoin de 11 mètres cubes, ce qui montre la taille du défi rencontré par ses constructeurs.

Pendant le fonctionnement du détecteur, les particules électromagnétiques créent des gerbes dans les cristaux qu'elles traversent. L'espace entre les cristaux doit être réduit au minimum pour éviter les fuites d'énergie. En conséquence, il a fallu concevoir une structure de soutien mécanique du ECAL plus mince que 0,5 millimètres, ce qui sera un tour de force quand le détecteur sera construit, car le poids total de cristaux sera de 80 tonnes.

Les cristaux de tungstate de plomb convertissent les gerbes électromagnétiques en lumière. ECAL emploiera des photodiodes à avalanche (APD) pour convertir cette lumière en signaux électriques proportionnels. Chaque cristal du ECAL aura une paire d'APD collé à l'arrière. Celles-ci feront partie d'un ensemble appelé capsule. La figure 2 montre une capsule et ses deux APD.

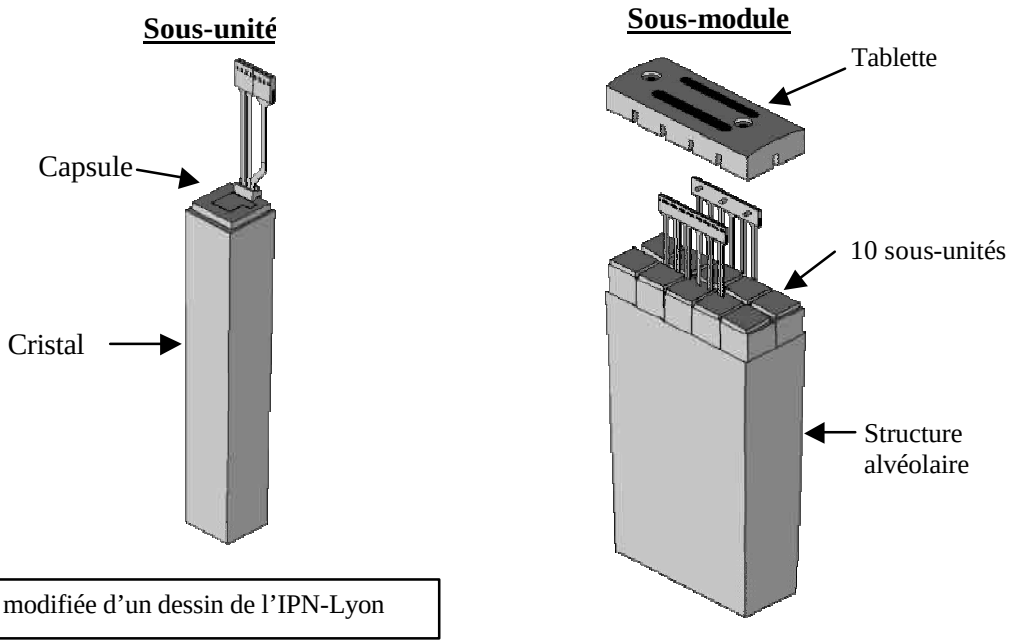



Version modifiée d'un dessin de l'IPN Lyon

**Figure 2 Vue éclatée d'une capsule**

Chaque cristal exige une paire d'APD parce qu'il n'a pas été possible de fabriquer des APD assez grands pour couvrir la superficie entière de l'extrémité d'un cristal. Une paire d'APD sera reliée à une chaîne d'électronique qui numérise et enregistrent leurs signaux électriques. L'APD et l'électronique doivent être de la précision la plus élevée pour ne pas altérer la précision de mesure permise par les cristaux. Une capsule plus un cristal s'appelle une sous-unité.

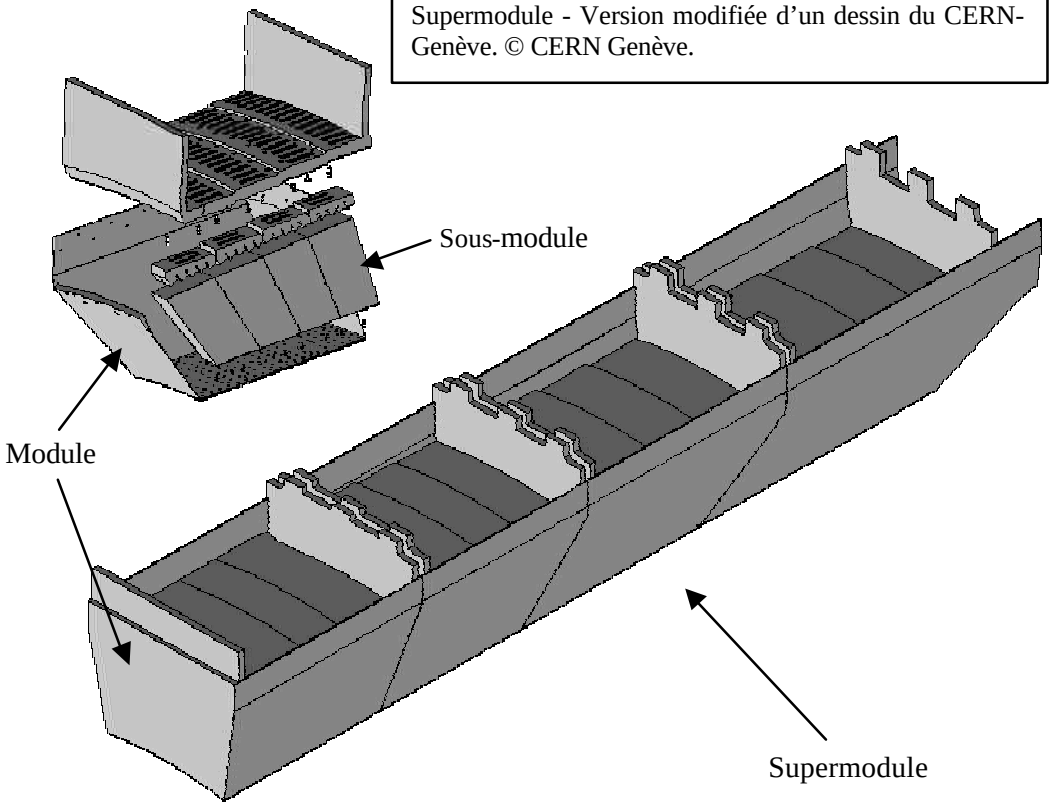
Les sous-unités vont ensemble former des sous-modules, comme représenté sur la figure 3.




Version modifiée d'un dessin de l'IPN-Lyon

**Figure 3 Une sous-unité et un sous-module.**

La figure 4 montre que les sous-modules sont assemblés en modules qui à leur tour forment des supermodules.



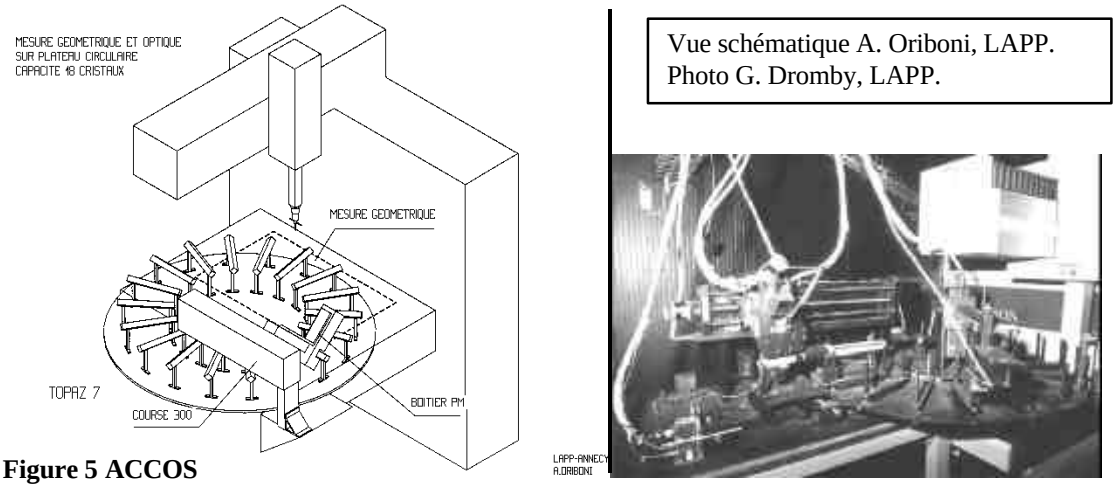

Module - Version modifiée d'un dessin de l'IPN-Lyon
Supermodule - Version modifiée d'un dessin du CERN-Genève. © CERN Genève.

**Figure 4 Module et supermodule**

En conclusion, les supermodules sont assemblés comme les segments d'une orange pour former le détecteur.

## *2.3 Construction des Sous-modules*

Après qu'un cristal ait été entré dans la chaîne de construction du ECAL, il est visuellement examiné, puis on mesure les dimensions spatiales, le rendement lumineux et la transmission longitudinale. Chacune des trois caractérisations est effectuée par un instrument – le Système Automatique de Mesure des Cristaux (Automatic Crystal Control System ACCOS) construit par le LAPP. La figure 5 montre une vue et une photographie schématiques d'Accos.




Vue schématique A. Oriboni, LAPP.
Photo G. Dromby, LAPP.

**Figure 5 ACCOS**

Les caractéristiques mesurées par ACCOS sont utilisées:

- Immédiatement pour détecter la non-conformité des cristaux.
- Dans la phase de construction pour déterminer la position optimale du cristal dans le détecteur (tous les cristaux sont différents, même avec le même type).
- Pendant la phase d'exécution du détecteur pour calibrer et faire le suivi du détecteur.

Les cristaux sortant de ACCOS sont triés par groupes de 10, représentant les futurs sous-modules. Le tri est basé sur le type et les caractéristiques de chaque cristal. Les groupes de cristaux reçoivent ensuite des capsules qui sont collées aux cristaux pour former des sous-unités. Chaque groupe est alors monté dans une structure alvéolaire et une tablette est placée sur le dessus pour former un sous-module.

La production des capsules commence par la réception des APD. Une série d'approximativement 2000 à 3000 APD est produite chaque mois, dont 2% sont envoyés par des essais de résistance au rayonnement. S'ils satisfont aux essais, le reste du lot entrera dans la chaîne de construction du ECAL. On prélève encore 2% des APD pour des essais destructifs tels que le vieillissement à long terme; voir [Patel et al, 1999] pour plus d'information sur l'assurance et le contrôle de qualité des APD. Aucun des APD n'est identique à un autre, d'où le besoin d'appariement. Le critère utilisé est la tension de polarisation qu'une APD exige pour fonctionner à un gain de 50. Une tolérance est définie pour cette tension de polarisation, et deux APD sont appariés s'ils satisfont tous deux œtte tolérance. Les 10 capsules qui composent un sous-module sont également appariées sur la base de la tension de polarisation qu'elles exigent pour fonctionner à un gain de 50. Ceci est exigé au niveau d'un sous-module, car un sous-module a une alimentation en tension unique pour tous les APD.
Le processus de construction des sous-modules est distribué dans le monde entier. Les cristaux sont produits en Russie, les sous-unités et les sous-modules sont assemblés en

Suisse et en Italie. Les APD sont produits au Japon, les tests de radiation sont effectués en Amérique, d'autres essais destructifs sont effectués en Suisse. Les capsules sont assemblées et testées en France

## *2.4 Particularités de la construction du ECAL*

La construction du ECAL est un apprentissage parce que le ECAL est unique. Les composants du détecteur et les procédés d'assemblage et de mesure qui leur sont appliqués évolueront pendant la phase de construction du détecteur. Pour réduire au minimum les coûts, le nombre de composants du détecteur rejetés pendant la construction doit être minimum. Un maximum de composants du détecteur doivent demeurer dans la production quand une nouvelle version des composants ou du processus de construction est appliquée. Certains composants pourront intégrer la nouvelle version et d'autres non. Si un composant ne peut pas intégrer la nouvelle version et s'il est encore acceptable dans la présente version, alors il doit continuer la chaîne dans cette version. Par conséquent à tout moment de la construction il peut y avoir deux composants ou plus du détecteur du même type suivant différentes versions du processus.

Les caractéristiques des différents composants du détecteur mesuré pendant la construction du détecteur seront utilisées en aval dans la phase de construction et pour le fonctionnement du détecteur. Un exemple a été donné dans la section précédente sous forme de caractéristiques des cristaux mesurées par ACCOS. On doit noter que quand cette thèse parle de composants du détecteur, cela se rapporte seulement à des composants dont les différentes caractéristiques sont exigées pour l'étalonnage du détecteur. Les écrous, les boulons et les rondelles ne sont pas considérés comme composants du détecteur.

L'entretien du détecteur peut seulement être effectué une fois par an. Quand il peut être fait, il est difficile d'accéder aux composants du détecteur et il y a le problème du niveau de radiation. Il est donc nécessaire d'enregistrer l'historique du procédé de construction de chaque composant individuel du détecteur pendant la construction. Cet historique est exigé pour l'entretien du détecteur, car il permet à des composants posant problème d'être localisés sans qu'il soit besoin de démonter et de tester les composants.

Comme on l'a montré dans la section précédente, la construction du ECAL sera répartie dans le monde entier. Les données représentant les caractéristiques et l'historique d'un composant individuel du détecteur devront suivre ce composant partout où elles seront expédiées.

## *2.5 Conclusion*

La construction du ECAL sera unique, répartie dans le monde entier, aura lieu sur une longue période (de 1999 à 2005), et produira une grande quantité de données (environ 1 Terabyte) sous forme des différentes caractéristiques et historique de construction des composants. Un système pour assister cette construction doit:

1. Donner aux constructeurs une assistance automatisée, parce qu'ECAL se compose de milliers de composants et sera construit sur une durée de 6 ans.
2. Enregistrer l'historique d'assemblage des différents composants du détecteur. L'entretien du détecteur est difficile et ne peut être exécuté qu'une fois par an. Connaître l'historique d'assemblage de chaque de composant aidera les utilisateurs à trouver les problèmes sans qu'il soit nécessaire d'accéder au détecteur.
3. Permettre à n'importe quel moment, à deux composants ou plus du même type, de suivre différentes versions du processus de conception et de construction.
4. Déterminer la capacité de production en temps réel pour permettre la planification de la construction avec un procédé en constante évolution.
5. Assurer un stockage de données centralisé pour la construction distribuée du ECAL. Les caractéristiques et l'historique d'assemblage des composants du détecteur sont générées dans les centres de fabrication distribués dans le monde entier. Cette information devra être rassemblée et stockée dans une base de données centrale où elle peut être recherchée pour l'étalonnage et l'entretien du détecteur.
6. Évoluer facilement, parce que de nouvelles fonctionnalités seront demandées système avec l'évolution du procédé de construction du ECAL.
7. dialoguer avec des systèmes qui n'existent pas encore ou qui sont actuellement inconnus aux réalisateurs de système, parce que pendant les phases de construction et d'exploitation du ECAL on prévoit que de nouveaux outils logiciels seront développés pour mesurer des caractéristiques et des historiques des composants.

Les conditions énumérées ne sont pas spécifiques à la construction du ECAL, mais sont également applicables à tout grand détecteur de physique. Par exemple, ces points sont applicables aux autres sous-détecteurs de l'expérience CMS. Les trois premières conditions démontrent le besoin de saisie de données contrôlée pendant la construction du détecteur, et la troisième condition démontre la nécessité de mesurer la capacité de production. Les trois dernières conditions indiquent que le système résultant devrait être distribué, évolutif et ouvert.

# 3 L'utilisation des méthodes de gestion de production au CERN

## *3.1 Introduction*

Le chapitre précédent a discuté le besoin de saisie de données contrôlée pendant la construction du détecteur, et la nécessité de mesurer la capacité de production. Ce chapitre décrit ce qui existe déjà au CERN. Ce chapitre commence par deux systèmes de gestion de production: La gestion de données des produits (PDM) et la gestion du déroulement des tâches (WfM), et discute comment ils sont actuellement utilisés au CERN. Le chapitre conclut sur la nécessité de développer un nouveau système de construction assistée par ordinateur pour aider à la construction du ECAL.

## *3.2 Gestion de données de produits*

La gestion de données de produit (PDM) est la gestion des données produites pendant le cycle de vie entier d'un produit, depuis la conception jusqu'à l'utilisation et la maintenance, et parfois jusqu'à la mise hors service. Le PDM fournit un paradigme axé sur le produit pour l'utilisation des données par les constructeurs, par opposition à la plupart des systèmes de gestion des données qui automatisent seulement les systèmes manuels existants pour accomplir les mêmes tâches plus vite [Williams, 1994]. L'approche axée par produit signifie que n'importe quel type de donnée ou document concernant un produit peut être obtenu simplement en indiquant l'identificateur du produit et le type d'information recherchée. Il n'y a plus besoin de rechercher les informations dans des systèmes de documentation séparés tels que les systèmes de conception et ceux de production. Les fonctionnalités actuelles des systèmes PDM sont récapitulées dans [ICMcData, 1995] comme:

- Stockage de données et gestion de documents
- Gestion de processus et de tâches
- Gestion de structure du produit (config.)
- Gestion des pièces et des composants (classification)
- Gestion de programme
- Communication et notification
- Transport des données
- Traduction des données
- Gestion d'images
- Administration système

Les systèmes actuels de PDM sont conçus pour faire face à des centaines de types de produit par opposition à des milliers d'éléments individuels. La saisie manuelle de quelques centaines d'éléments est acceptable, mais la saisie manuelle de milliers d'éléments sans automatisation est tout simplement impossible, et source sûre d'erreurs humaines. Les systèmes actuels de PDM ne fournissent pas une telle automatisation.

## *3.3 Gestion de processus*

La Coalition de Gestion de Processus (WfMC) [Hollingsworth, 1995] définit le déroulement des tâches comme "facilitation ou automatisation informatisée d'un processus industriel, en entier ou en partie" et un système de gestion de processus comme "un système qui définit complètement, contrôle et exécute un processus par l'intermédiaire d'un logiciel qui pilote l'exécution par une représentation informatisée de la logique de déroulement des opérations". Les systèmes de gestion de processus permettent à des procédés de travail d'être modélisés,

analysé et améliorés. Ils permettent également à des procédés de travail d'être automatisés et contrôlés. Cette thèse distingue deux types de système de gestion de processus. Ceux qui gèrent les versions des documents qui décrivent un produit, et ceux qui gèrent le cycle d'évolution des différents composants qui constituent un produit.

Les systèmes actuels de gestion de processus ont été conçus pour et utilisés dans des environnements où les processus industriels sont bien connus et bien compris. Un scénario typique étant une compagnie qui est passée par un programme industriel de réorganisation. Des processus informels existants sont transformés et enregistrés sous forme logique et informatique telle qu'un système de gestion de processus. Cet environnement n'est pas sujet à évolution, vu que le processus est déjà bien connu. Par conséquent les techniques de gestion actuelles de processus ne supportent pas bien l'environnement de « production unique » d'un grand détecteur de physique où les processus évoluent pendant la construction.

## *3.4 Techniques de gestion de production utilisées CERN*

Le CERN a un projet de gestion de données des produits appelé « **C**ERN **E**ngineering data management system for **D**etectors and **A**ccelerato**R** » (CEDAR). Ce projet a été lancé pour contrôler les données de conception, de construction, d'installation et d'entretien du futur Grand Accélérateur de Hadrons (LHC) et de ses détecteurs. Le projet comprend deux parties : Le fonctionnement d'un système industriel de PDM appelé CADIM/EDB (Computer Aided Data Integration Manager / Engineering Data Base) et le fonctionnement de son interface Web - TuoviWDM, voir [Muller et Widegren, 1998] pour l'interaction avec l'utilisateur. Sans compter le projet CEDAR, le CERN utilise également un outil de gestion de documents appelé « CERN Drawing Directory » (CDD), voir le manuel de l'utilisateur [Delamare et Petit, 1998] pour plus d'information. La figure 6 montre les rapports entre les différents outils de gestion de production utilisés au CERN:

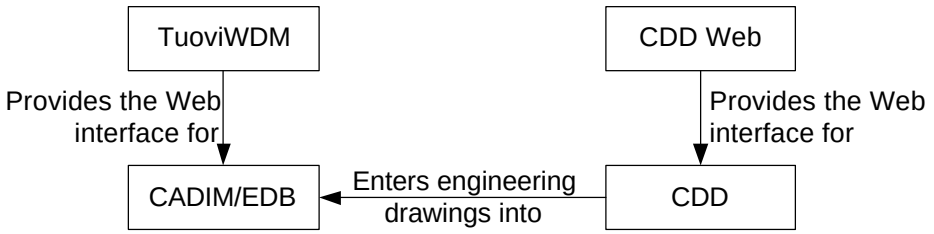



**Figure 6 Outils de gestion de production utilisés au CERN**

TuoviWDM fournit l'interface Web pour CADIM/EDB et de même CDDWeb fournit l'interface Web pour CDD. Des schémas d'ingénierie peuvent être entrés dans CADIM/EDB en utilisant CDD.

Pour récapituler, le logiciel actuel du projet CEDAR permet d'accéder aux documents qui décrivent un détecteur et de les visualiser. Des techniques de gestion de processus sont employées pour aider des utilisateurs à contrôler le cycle d'évolution des documents.

## *3.5 Conclusion*

Quoique le projet CEDAR soit ciblé sur l'ensemble de la vie du détecteur, son orientation document n'est pas compatible avec les besoins identifiés dans le chapitre précédent. Des techniques de gestion de document ont été développées dans l'industrie principalement pour la

conception et les plans détaillés de grands projets d'ingénierie. Le CERN les a correctement utilisés pendant ces phases, toutefois nous avons montré dans cette thèse qu'elles ne peuvent pas être appliquées avec succès à la phase de construction. Il y a deux raisons possibles pour lesquelles aucune recherche n'a été faite dans ce domaine avant. Tout d’abord, la complexité et surtout la durée de vie des détecteurs au CERN ne justifiaient pas une analyse intensive de la construction du détecteur. Historiquement, des solutions ad-hoc ont été préférées et cela se justifie. Ensuite, les physiciens et les ingénieurs responsables de la construction du détecteur n'ont pas été au contact des méthodes de génie industriel utilisées pendant la conception. Par conséquent leur connaissance des techniques de gestion de production industrielle a été limitée. Les conditions ont maintenant changé avec l'arrivée de la prochaine génération de détecteurs au CERN. Ils seront dix fois plus complexes que la génération précédente et nécessiteront de plus longues périodes de construction. Les solutions ad-hoc ne seront plus viables. Il est donc nécessaire de développer un nouveau système de construction assisté par ordinateur qui soit orienté vers les composants physiques du détecteur et non vers les documents qui les décrivent. Un système d’assistance par ordinateur, prévu pour le long terme, évolutif et ouvert de support de construction est indispensable pour remplir les conditions de la construction du ECAL présentées dans le chapitre précédent. Cette thèse apporte une contribution développement de CRISTAL, le premier système de support de construction à l'usage de n'importe quel détecteur, qui suit les composants individuels du détecteur. Le prochain chapitre présente CRISTAL et son architecture distribuée.

# 4 Une architecture distribuée pour la construction d'un détecteur en coopération mondiale

## *4.1 Introduction*

Le chapitre 2 a démontré qu'un système d'aide à la construction du détecteur ECAL en coopération mondiale doit avoir un système de stockage de données centralisé. Ce chapitre présente l'architecture de CRISTAL (Co-operative Repositories and Information System for Tracking Assembly Lifecycle). Le but principal de l'architecture est de faire face à la répartition mondiale de la construction du détecteur. L'architecture est distribuée avec un système central contrôlant un ou plusieurs centres de fabrication situés dans le monde entier. Le chapitre expose d'abord la portée et les objectifs de CRISTAL, puis fait une description de ses utilisateurs. L'architecture est alors présentée dans 3 sous-sections: l'architecture globale, le système central, et les centres de fabrication. Le chapitre conclut en présentant les problèmes que l'architecture permet de surmonter.

## *4.2 Objectifs et fonctions de CRISTAL*

CRISTAL est un système logiciel destiné à faciliter la construction d'un grand détecteur de physique. Les deux objectifs principaux de CRISTAL sont de recueillir toutes les données de construction exigées pour l'étalonnage et l'entretien du détecteur, et de fournir une gestion orientée composants du processus de construction du détecteur.

La figure 7 montre où CRISTAL est utilisé dans le cycle de vie d'un grand détecteur de physique.

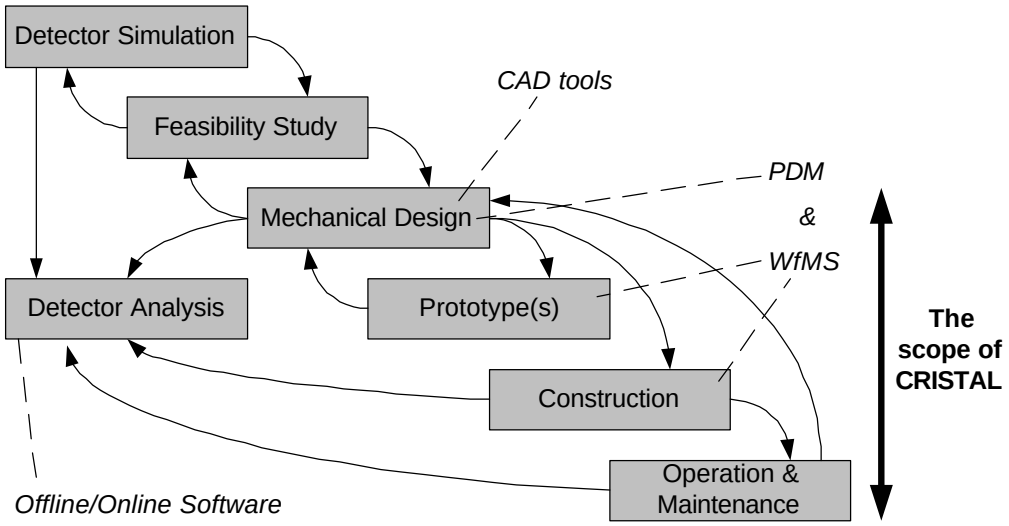



**Figure 7 CRISTAL dans le cycle de vie d'un détecteur de physique**

Avant même le commencement de la conception mécanique d'un détecteur, les physiciens effectuent des simulations et des études de faisabilité. Si le détecteur est accepté, les concepteurs utilisent des outils de conception assistée par ordinateur (CAO) pour produire les nombreux schémas nécessaires pour décrire ce que sera le détecteur à et de quoi il se composera. Le logiciel de gestion de données des produits (PDM) est utilisé pour traiter le grand nombre de documents. CRISTAL fournit le support de gestion de processus pendant la phase de construction du détecteur. Cette aide est axée sur le composant physique et ne doit

pas être confondu avec les systèmes de gestion de processus dits PDM, qui contrôlent le cycle de vie des documents qui décrivent les composants par opposition aux composants eux-mêmes.

CRISTAL est axé sur les composants. Ceci signifie qu'un utilisateur dit au système avec quel composant individuel du détecteur ils souhaitent travailler. Puis en fonction de celui-ci on effectue les activités d'assemblage et de mesure dans le contexte de ce composant. La liste suivante présente ce que CRISTAL doit faire pour chaque composant du détecteur:

- Afficher les prochaines tâches possibles dans le processus de construction,
- enregistrer et tester le résultat de chaque tâche,
- transférer automatiquement les instructions aux instruments utilisés dans le procédé de construction et enregistrer les résultats,
- enregistrer l'historique de la construction,
- proposer des composants fils et parents possibles pour l'assemblage,
- saisir l'emplacement dans le détecteur,
- fournir un accès contrôlé à toutes les données de construction saisies.

### *4.3 Utilisateurs de CRISTAL*

CRISTAL groupe les utilisateurs selon leur rôle. Le tableau 1 détaille ces rôles et leur description.

| Rôle utilisateur | Description |
|---|---|
| Information MANager (IMAN) | Il y a au moins un IMAN pour le système central et un IMAN à chaque centre de production. IMAN est responsable de l'installation et l'entretien du système de CRISTAL, y compris la gestion des utilisateurs de CRISTAL et des bases de données de CRISTAL |
| COORdinateur (COOR) | Il y a un COOR au système central. Il est responsable du suivi du schéma de production qui indique la composition du détecteur et des processus qui contrôlent sa construction |
| OPERateur (OPER) | Un ou plusieurs OPER travaillent dans un centre de production. Ils exécutent des activités de mesure et assemblent des composants du détecteur. |
| Centre SUPerviseur (CSUP) | Un ou plusieurs CSUP travaillent dans un centre de production. Un CSUP peut faire tout qu'un OPER peut faire et en plus il peut:<br>• Rejeter un composant défectueux du processus de construction<br>• Permettre aux composants qui n'ont pas le niveau exigé de qualité de continuer dans le processus de construction comme "non-conforme"<br>• Définir les droits d'accès des données de construction comme disponible à chacun ou seulement à un centre de fabrication. |
| PHYSicist (PHYS) | PHYS étudie les caractéristiques de différents composants du détecteur dans CRISTAL. Il a besoin de moyens faciles pour indiquer et obtenir rapidement ce qu'il veut parmi le Terabyte prévu de données de construction. Beaucoup de physiciens ne travailleront pas dans un des centres de CRISTAL. Leur interface au système de CRISTAL doit donc être indépendante de leur localisation. |

**Table 1 Rôles des utilisateurs de CRISTAL**

### *4.4 Architecture globale de CRISTAL*

CRISTAL modélise la construction distribuée d'un détecteur de physique comme un système central contrôlant un ou plusieurs centres de fabrication répartis dans le monde entier. Le système de contrôle de production doit s'assurer que chaque centre de fabrication est assez autonome pour continuer la production quand la connexion avec le monde extérieur est cassée. Ceci a deux conséquences :

- Un centre de fabrication doit stocker assez d'information pour permettre à la production locale de continuer alors que la connexion avec le monde extérieur est interrompue.
- Les données doivent être stockées localement et sécurisées quand la connexion au monde extérieur est cassée et transmises plus tard quand la connexion est réparée.

La figure 8 montre l'architecture globale de CRISTAL.

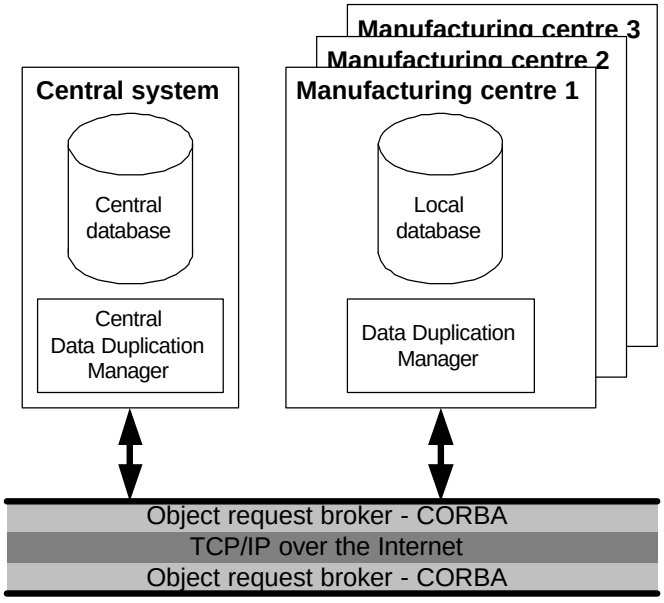



**Figure 8 Architecture globale de CRISTAL**

Le système central est relié aux centres de production via Internet en utilisant un « Object Request Broker » ou courtier de demande d'objet (ORB). Un ORB est un bus logiciel qui contrôle la transmission et l'échange de données entre les objets. CRISTAL utilise l'Architecture Commune de courtier de demande d'objet (CORBA) du groupe de gestion d'objet (OMG), voir [OMG, 1999] pour plus de détails. CORBA est indépendant du langage et de la plateforme de programmation. C'est une norme ouverte, ce qui signifie qu'il qu'elle ne dépend d'aucune compagnie. Les interfaces des objets distribués sont définies dans un langage standard de définition d'interface (IDL) et puis compilées pour produire le code dépendant du langage et de la plateforme.

Chaque centre de fabrication a sa propre base de données locale pour stocker l'information exigée pour travailler de façon autonome sans connexion extérieure. Le système central a une grande base de données qui enregistre un double de toutes les données de production recueillies aux centres de fabrication. La taille prévue de cette base de données est approximativement 1 Terabyte.

Le système central et chaque centre de production a un gestionnaire de duplication de données responsable du stockage des données quand elles ne peuvent pas être envoyées au monde extérieur, et de les envoyer plus tard quand la connexion est rétablie.

### *4.5 Architecture du système central*

Le système central est employé pour fournir le contrôle global de construction du détecteur et pour recueillir toutes les données de construction dans un seul endroit. La figure 9 montre l'architecture centrale du système:

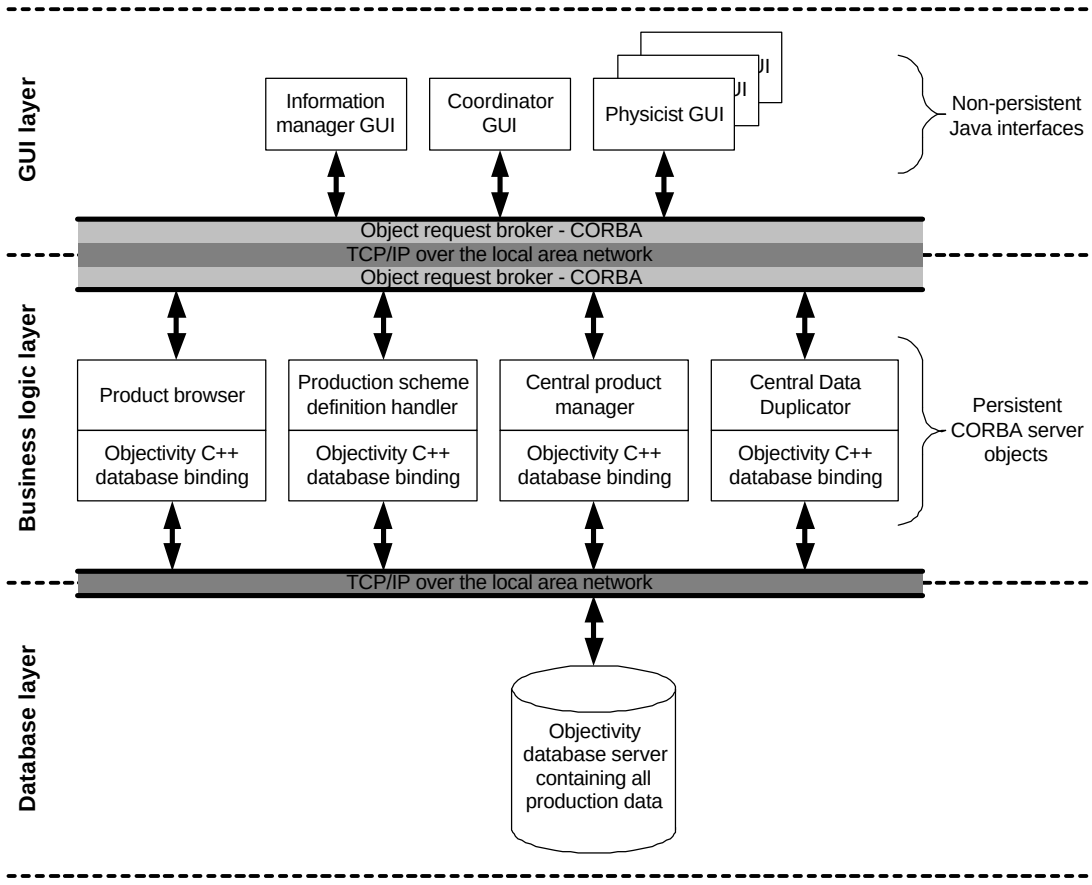



**Figure 9 Architecture du Système Central**

L'architecture est un système distribué typique à 3 niveaux. La couche supérieure fournit les interfaces utilisateur graphiques (GUI), la suivante fournit la logique d'entreprise, et la dernière fournit la gestion de base de données. Toutes les interfaces CRISTAL sont écrites en Java pour la portabilité, toute la logique d'entreprise est écrite en C++ pour l'efficacité et le logiciel Objectivity fournit la fonctionnalité de base de données. Les GUI communiquent avec la couche logique par CORBA, et la logique d'entreprise utilise un module Objectivity C++ pour accéder au système de gestion de base de données. Seule la couche de logique d'affaires est persistante, les GUI de CRISTAL n'ont aucune persistance.

Le système central utilise des GUI spécialisés pour le gestionnaire d'information, le coordinateur COOR et le physicien. La logique d'entreprise est divisée en quatre objets serveur CORBA:

1. Afficheur de produit. Avec le GUI physicien, permet la recherche des données de production parmi tout ce qui a été rassemblé au système central.

2. Gestionnaire de définition de schéma de production. Une définition de schéma de production indique la composition "à-construire" du détecteur et l'ordre des activités qui doivent être exécutées pour le construire. Avec le GUI de coordinateur, le Gestionnaire de définition de schéma de production permet la création de nouveaux schémas de production.
3. Gestionnaire central de produit. Pour chaque composant du détecteur dans un centre de fabrication, il y a un objet œrrespondant du logiciel appelé un "gestionnaire de produit". Cet objet contient la logique de production afin de gérer l'assemblage du composant, fournir le support de gestion de processus et saisir les données de construction. Un gestionnaire de produit crée un « événement » (une note informatique) chaque fois qu'il met à jour la base de données du centre de fabrication. Ces événements sont reproduits et envoyés au système central. Le gestionnaire central de produit reçoit tous les événements produits par tous les chefs de produit dans tous les centres de fabrication. Il emploie ces événements pour recréer les données de construction de chaque centre de fabrication dans le système central.
4. Gestionnaire central de duplication. Pour rendre transparents les problèmes de connexions interrompues avec le monde extérieur.

### *4.6 Architecture des Centres de Production*

L'architecture de centre de production est semblable à celle du système central en trois niveaux , il y a un GUI pour le gestionnaire d'information et le physicien, et un objet CORBA serveur de l'afficheur de produit. Il y a un GUI spécifique au centre de fabrication pour les opérateurs de construction sont attribués aux composants du détecteur, exécutent des processus et saisissent des données de construction. La figure 10 montre l'architecture du système CRISTAL un centre de fabrication.

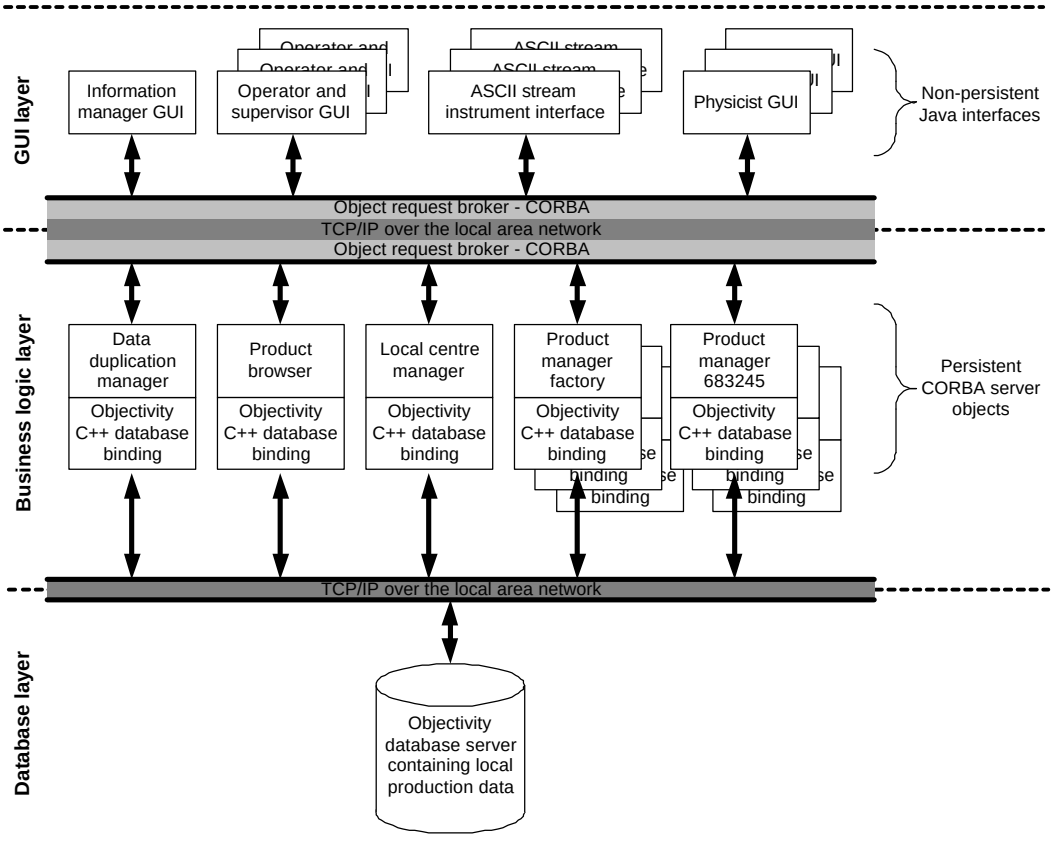



**Figure 10 Construction de l'architecture centrale**

Il y a quatre objets serveur CORBA dans la couche de logique d'entreprise qui sont spécifiques à l'architecture d'un centre de fabrication:

1. Gestionnaire de centre local. Connaît quels produits sont actuellement au centre et dans quel état. Contrôle les listes d'utilisateurs. Supervise l'assemblage des composants, et répartit les tâches de calcul de CRISTAL sur les ordinateurs disponibles au centre de fabrication.
2. Chef de produit. Représente un composant individuel du détecteur. Les chefs de produit avec le GUI "opérateur et superviseur" permettent à des utilisateurs d'exécuter des tâches sur différents composants du détecteur et de les assembler.
3. Atelier de chef de produit. Il y a dans chaque ordinateur un « atelier de chef de produit » utilisé pour activer des objets chef de produit. Ces ateliers créent des objets « chef de produit » sur demande de l'objet local gestionnaire de centre.
4. Gestionnaire de duplication de données. Stocke les données à envoyer au monde extérieur quand la connexion est cassée, et les envoie quand la connexion est rétablie.

## *4.7 Conclusion*

Le but principal de l'architecture de CRISTAL est de rassembler et de stocker dans un système d'archivage central les caractéristiques et l'historique d'assemblage des composants du détecteur provenant des centres de production répartis à l'échelle mondiale. Ce chapitre a prouvé que l'architecture adoptée résout les problèmes de transfert et d'enregistrement des données dans un ensemble distribué de centres de fabrication, et que l'architecture permet de définir les rôles des utilisateurs et les points où ils doivent interagir avec le système.

# 5 Représentation des processus

La phase de construction d'un grand détecteur de physique implique des milliers de composants et dure pendant plusieurs années. Il est donc nécessaire de guider les constructeurs pendant cette phase. Le chapitre 2 a montré la nécessité de contrôler la saisie de données pendant la construction du détecteur. Pour aider à remplir cette condition CRISTAL définit une représentation des processus pour illustrer le cycle de vie d'un composant du détecteur pendant la construction du détecteur. La représentation est utilisée dans deux buts: pour indiquer l'ordre des activités à exécuter sur un type de composant du détecteur pendant la construction, et pour montrer la position actuelle d'un composant individuel dans sa propre séquence de construction. Ce chapitre décrit donc la représentation dans deux sections correspondantes.

## *5.1 Représentation des processus de CRISTAL*

Une définition de processus de CRISTAL décrit la séquence particulière des activités qui doivent être effectuées sur un composant du détecteur pendant la construction du détecteur. Un processus est dessiné sous forme de graphique acyclique[2] orienté des activités, qui peut se dédoubler et se joindre. La figure 11 donne un exemple:

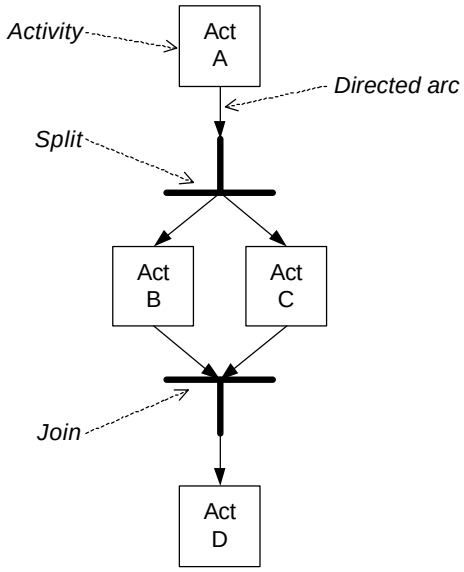



**Figure 11 Graphe de définition de processus**

Les définitions de processus sont lues de haut en bas. La figure 11 indique qu'il faut exécuter l'activité A, puis B et C dans n'importe quel ordre, et puis D quand B et C sont terminés. Les divisions et les jonctions permettent le parallélisme. On peut exécuter les activités B et C dans n'importe quel ordre, ou en parallèle. Le parallélisme est exigé parce que quelques grands composants du détecteur auront beaucoup de tâches effectuées sur eux en même temps. Il faut noter que le processus qui suit une jonction ne peut pas être commencé avant que toutes les tâches précédant la jonction soient terminées.

[2] Un ensemble de nœuds reliés par des arcs orientés, avec la contrainte qu'il n'y a pas de boucles.

Un graphique valide de définition de processus commence par une activité ou une division, et se termine avec une activité ou une jonction. Chaque division doit correspondre à une jonction. La figure 12 illustre ceci en montrant deux combinaisons, une valide et une incorrecte des divisions et des jonctions.

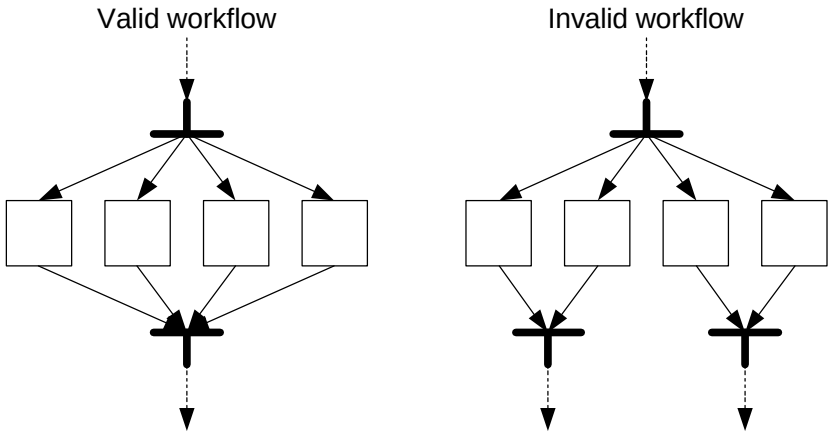



**Figure 12 Combinaisons de divisions/jonctions valides et invalides**

Le processus du côté gauche est valide car la division en haut du diagramme appariée avec la jonction au bas. Le processus du côté droit est incorrect parce que le nombre de divisions n'est pas égal au nombre de jonctions.

La représentation de processus de CRISTAL utilise l'idée d'activités composites pour l'emboîtement des définitions de processus. L'emboîtement réduit la complexité visuelle, et permet à des sous-sections d'un procédé de construction d'être marquées dans un but de gestion. L'utilisateur qui commence une activité composite devient automatiquement le gestionnaire de toutes ses sous-activités. Ceci signifie que l'utilisateur peut exécuter les actions suivantes sur les sous-activités: marquer une activité terminée pour être répétée, remettre dans l'état initial une activité en cours d'exécution, et sauter une activité pas encore commencée.

Un graphique de définition de processus sans état est employé pour indiquer l'ordre des activités à exécuter sur un ou plusieurs types de composants du détecteur. Pendant la construction du détecteur, différents composants physiques doivent être traités indépendamment en ce qui concerne leur place dans la séquence de construction. Des composants individuels de même type du détecteur peuvent suivre différentes versions du processus de construction. Ceci signifie qu'un agent constructeur ne peut pas dire quelle est la prochaine activité à exécuter sur un composant simplement à partir de sa place dans la chaîne de production. Les constructeurs doivent savoir la position de chaque composant individuel du détecteur dans sa propre version du procédé de construction. CRISTAL visualise la position d'un composant dans son cycle de vie de construction en ajoutant l'information d'état à un graphique de définition de déroulement des opérations. La prochaine section décrit la notation employée pour représenter cet état.

## *5.2 Définitions de processus de CRISTAL avec état*

À tout moment pendant le cycle de construction d'un composant du détecteur, quelques activités auront été terminées, certains auront été commencés et d'autres attendront encore pour être commencées. À la fin du cycle toutes les activités auront été terminées. Ceci indique qu'une activité de construction a un état et un comportement dynamique. CRISTAL visualise la position actuelle d'un composant dans sa propre version du processus de construction en affichant l'état actuel de chaque activité sur un graphique de définition du processus. Avant de présenter la notation employée pour représenter l'état d'activité, il est nécessaire de décrire en détail le comportement dynamique d'une activité de construction. La figure 13 est un réseau de transition d'états finis (FSTN) montrant une version simplifiée de ce comportement.

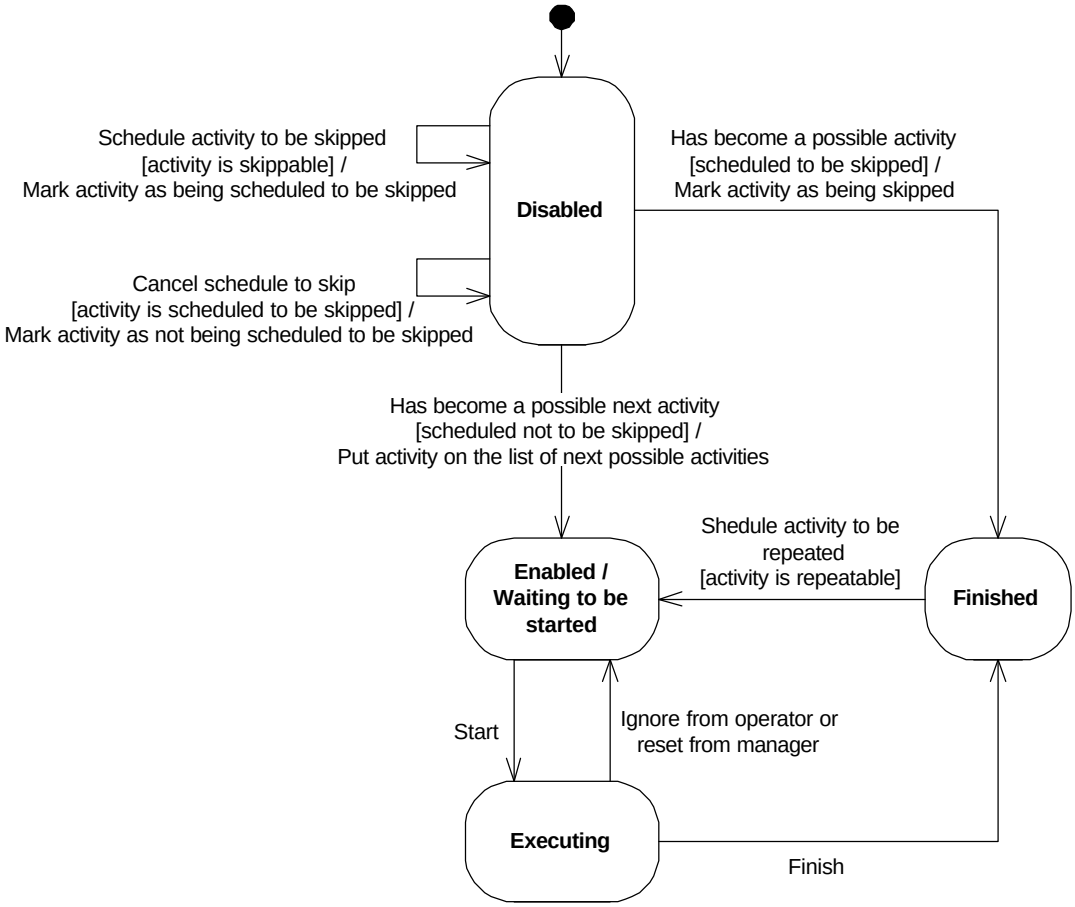



**Figure 13 Comportement dynamique simplifié d'une activité de construction**

Le FSTN a quatre registres booléens non montrés sur la figure: "qu'on peut répéter", "qu'on peut sauter", "programmé pour être sauté" et "sauté". Une activité démarre dans l'état désactivé avec les registres "programmé pour être sauté" et "saut" dans l'état « faux ». Dans cet état l'utilisateur peut programmer l'activité pour être sautée. Quand l'activité devient une prochaine activité possible elle entrera normalement dans l'état "en attente de démarrage". Cependant si l'activité est programmée pour être sautée, alors elle entrera directement dans l'état "terminé". A un certain point l'utilisateur commencera l'activité. Une fois démarrée l'activité entrera dans l'état "en cours d'exécution". A partir de là l'activité peut être ignorée, réinitialisée, ou terminée. Ignorer ou réinitialiser l'activité la renvoie à l'état "en attente de démarrage", tandis que terminer l'activité la met dans l'état "terminé". Une activité terminée

qu'on peut répéter peut être programmée pour être répétée, l'amenant dans l'état "en attente de démarrage".

La figure 15 montre la notation employée pour visualiser l'information d'état d'activité.

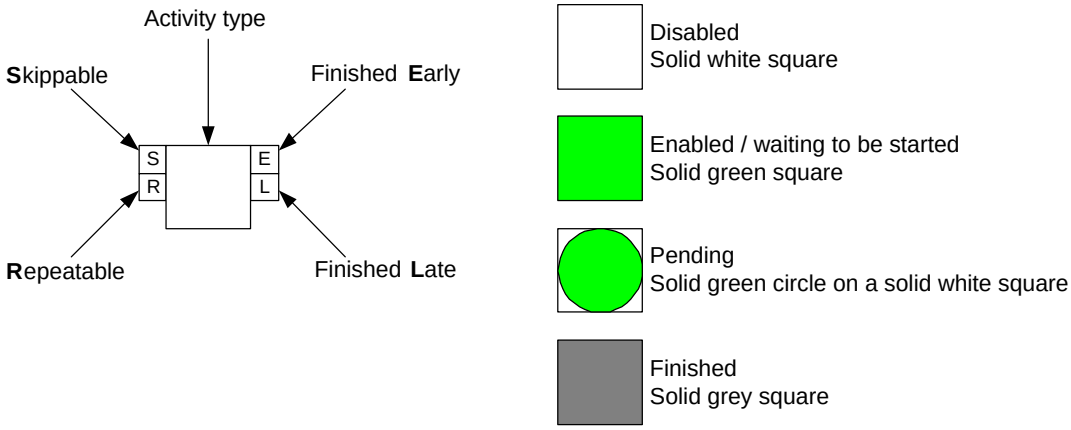



**Figure 14 Représentation de l'information de l'état d'activité**

Pour terminer la description du comportement dynamique d'une activité de construction il est nécessaire de déclarer que la plupart des activités de construction exécutées sur des composants génèrent un résultat. Ces résultats sont enregistrés quand les activités sont terminées, et ils vont permettre de définir les caractéristiques des différents composants du détecteur.

# 6 Une modélisation des dessins et des processus pour les composants du détecteur

## *6.1 Introduction*

Le chapitre 2 a montré qu'un système d'aide à la construction du détecteur doit permettre à n'importe quel moment à deux composants ou plus d'un même type de suivre différentes versions du processus de conception et de construction. Ce chapitre décrit le modèle de données de CRISTAL, qui répond à cette condition.

Le chapitre commence par présenter les exigences des utilisateurs que le modèle de données a dû résoudre. Il décrit alors pourquoi la configuration "de description d'élément" a été utilisée et où elle a été utilisée. Ceci est suivi d'une vue d'ensemble du modèle de données de CRISTAL, montrant comment la composition du détecteur a été intégrée avec les définitions de processus. Le mécanisme de version permettant de faire face à l'évolution du dessin et des processus liés aux composants du détecteur est alors décrit. Enfin le chapitre conclut en montrant que l'approche orientée description adoptée par le modèle de données de CRISTAL est une méthode appropriée pour décrire la composition de n'importe quel grand détecteur HEP et des processus qui commandent et contrôlent sa construction.

Ce chapitre utilise la notation du Langage Unifié de Modélisation (UML) parce que l'analyse et les phases de conception du modèle de données de CRISTAL ont été effectuées en utilisant une méthodologie orientée-objet et la notation UML[3].

## *6.2 Conditions utilisateur pour le modèle des données*

Le modèle de données de CRISTAL doit remplir les conditions suivantes des utilisateurs:

- La composition complète du détecteur et des processus employés pour contrôler sa construction ne sera pas connue jusqu'à ce que la production soit déjà en cours. Le modèle de données doit donc permettre d'ajouter des définitions de composition du détecteur et des processus pendant la production.

- La nature d'unicité de la construction du détecteur implique que les définitions des composants et les processus évolueront au fur et à mesure que de meilleurs dessins et des procédés de production plus efficaces sont trouvés. Par conséquent le modèle de données doit faciliter l'évolution des définitions.

- Chaque activité de production produit des résultats dont le format de données n'est pas connu lors de la compilation du programme. Le modèle de données doit permettre la création de nouveaux formats de données définis par l'utilisateur en temps réel.

[3] Les lecteurs peu familiers avec UML peuvent lire l'appendice C pour une courte introduction. Les lecteurs demandant une explication plus en profondeur peuvent se référer à [UML, 1997]

### *6.3 Pourquoi un modèle de « description d'élément »*

Les modèles d'analyse "de description d'élément" représentent les éléments et leurs descriptions en tant qu'objets séparés : un objet de description décrit un ou plusieurs objets d'élément. La figure 15 est un diagramme de classe paramétrisé, copié de [Blaha et al 1998], montrant la configuration ainsi qu'un exemple. Des paramètres sont dénotés avec des équerres, par exemple. "<paramètre>".

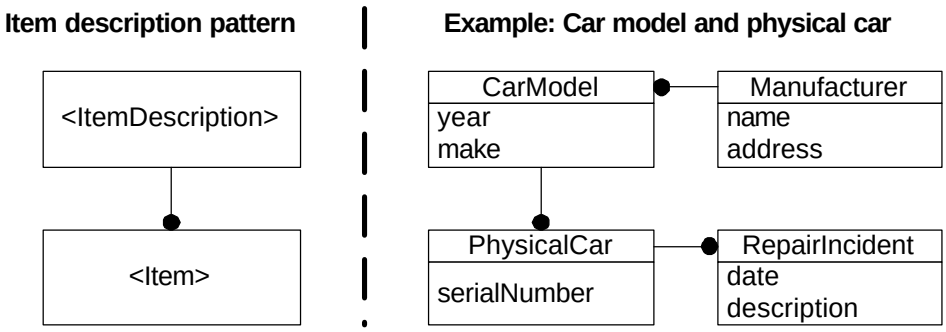



**Figure 15 Modèle de description d'élément [Blaha et al 1998]**

L'exemple montre qu'il y a beaucoup de voitures physiques du même modèle, et que l'année et le type sont spécifique au modèle, tandis que le numéro de série est spécifique à chaque voiture individuelle. L'exemple montre également les rapports avec les constructeurs de voiture et les incidents mécaniques. Un constructeur peut produire beaucoup de modèles de voiture, et une voiture physique peut avoir beaucoup d'incidents mécaniques.

L'utilisation du modèle de description d'élément entraîne l'utilisation d'une « information de description » qui se placerait normalement au niveau de la classe dans un modèle orienté-objet, mais ici se place au niveau de l'objet. Dans l'exemple de modélisation des composants du détecteur, au lieu d'employer une classe simple pour décrire ce qu'est un composant, le modèle de description d'élément utilise deux classes: un pour décrire ce qu'est une description d'un composant du détecteur, et un pour décrire ce qu'est un composant individuel du détecteur. En déplaçant les données de description du niveau de classe au niveau d'instance, on élimine pratiquement le besoin d'évolution de schéma de base de données. Un autre avantage important du modèle de description d'élément est que des objets d'élément sont faiblement couplés à leurs objets de description. Ceci donne un environnement idéal pour concevoir et mettre en application un mécanisme flexible pour des gérer des versions de composants et de processus attachés.

Le modèle de données de CRISTAL a adopté le modèle de description d'élément, parce que l'alternative serait de modéliser des descriptions au niveau de la classe, ayant pour résultat un nombre inacceptable d'évolutions de schéma, c'est à dire une évolution de schéma chaque fois qu'un utilisateur présente un nouveau type de composant du détecteur.

### *6.4 Utilisation du modèle de « description d'élément »*

Le tableau 3 montre l'utilisation de la configuration de description d'élément dans le modèle de données de CRISTAL. DAG signifie graphe acyclique[4] orienté.

| <Description d'élément> | Type de graphe | Signification | <Elément> | Type de graphe | Signification |
|---|---|---|---|---|---|
| Définition de produit | DAG | "A construire" Composition du détecteur | Produit Physique | Arbre | "construit" Composition du détecteur |
| Définition d'activité | DAG | Processus de composant | Activité | DAG | Etat de processus |
| Définition de données | DAG | Format défini par utilisateur pour résultat d'activité | Donnée | Arbre | Donnée résultat d'activité |

**Table 2 Utilisation du modèle de « description d'élément »**

La table montre que le modèle de description d'élément est utilisé en modélisant la composition en détecteur, les processus des composants et les données de résultats d'activité.

L'utilisation du modèle de description d'élément divise proprement l'architecture de CRISTAL en logiciel hors-ligne et en ligne. Le diagramme paramétrisé de classe suivant prouve que le logiciel hors-ligne crée et modifie des objets de description, tandis que le logiciel en ligne lit seulement des objets de description et crée et modifie des objets d'élément.

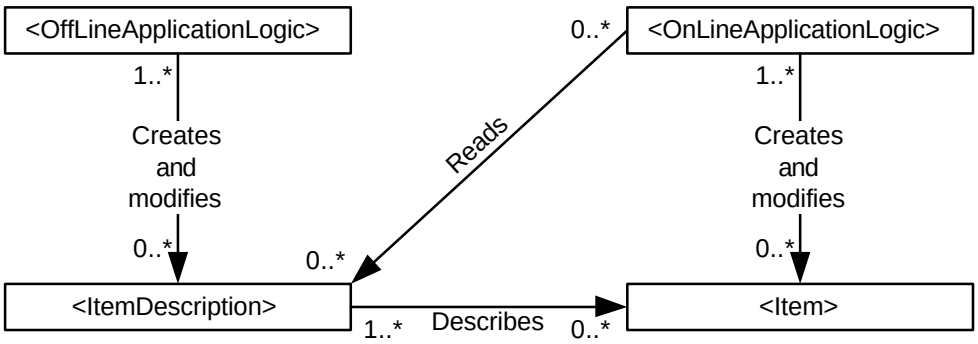



**Figure 16 Logiciel en-ligne et hors-ligne**

La division de l'architecture en logiciel hors-ligne et en ligne a facilité le développement. Une fois que les classes de description d'élément ont été finalisées, une équipe de programmeurs a travaillé sur la logique d'entreprise hors-ligne et sur les GUI indépendamment d'une autre équipe qui a travaillé à la logique d'entreprise et sur les GUI en ligne.

[4] Un ensemble de nœuds reliés par des arcs orientés et ne comportant pas de boucles.

Le tableau 4 montre les classes du système de CRISTAL divisées en logiciel hors-ligne et en ligne.

| | Hors-ligne | En-ligne |
|---|---|---|
| **Classes de GUI** | Interface Coordinateur | Interface opérateur |
| **Classes de logique d'entreprise** | Gestionnaire de définition de schéma de production | Gestionnaire de produit |
| **Classes de base de données** | Définition de Produit<br>Définition d'Activité<br>Définition de Données | Produit<br>Activité<br>Données |

**Table 3 Classes de Cristal hors-ligne et en-ligne**

Les classes de base de données représentent le modèle de données de CRISTAL. Les paragraphes suivants expliquent ce qui est spécifique à chaque classe de base de données.

### 6.4.1 Objet de Produit et Objet de Définition de Produit

La figure 18 montre les classes de produit et de définition de produits et les rapports entre elles.

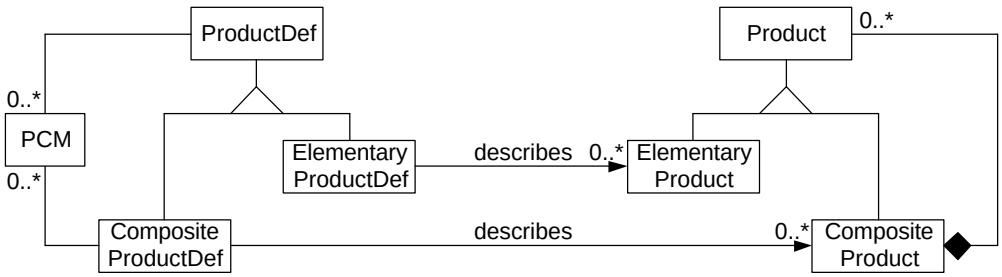



**Figure 17 Objet de Produit et Objet de Définition de Produit**

Chaque objet de définition de produits contient le détail de l'information propre à un type de composant du détecteur, y compris le nom et la description textuelle. Chaque objet de produit contient le détail de l'information relative à un composant individuel du détecteur, par exemple son identificateur unique (code à barres) et sa composition actuelle. Objets de produit et de définition de produits, comme ce qu'ils représentent, peuvent être composés ou élémentaires. Des objets de définition de produits sont reliés ensemble dans les graphiques acycliques dirigés pour représenter la structure "à-construire" du détecteur, et des objets de produit sont reliés ensemble dans les arbres pour représenter la structure "comme-construite" du détecteur. Les objets composés de définition de produits sont reliés à leurs membres par l'intermédiaire des objets du Membre de composition du produit (PCM) - un objet de PCM par membre. La figure 18 donne un exemple.

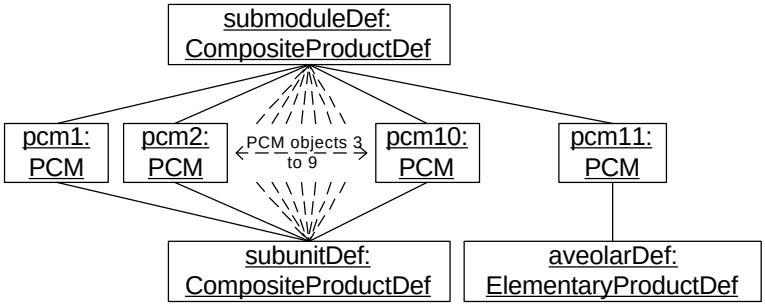



**Figure 18 Exemple en DAG d'objet de définition de produit**

Un sous-module se compose de 10 sous-unités et 1 structure alvéolaire. Par conséquent il y a 10 objets de PCM reliant l'objet de définition de sous-module à l'objet de définition de sous-unité, et 1 objet de PCM reliant l'objet de définition de sous-module à l'objet alvéolaire de définition. Sous leur forme la plus pure, les objets de PCM représentent les différents membres d'un produit composé. À l'heure actuelle ils sont employés pour représenter l'unique emplacement géométrique de chaque membre. Des conditions de début d'activité sont définies en termes de ces emplacements uniques. Pendant la production, quand un composant du détecteur est assigné à un emplacement à l'intérieur d'une unité plus grande, son seul identificateur - un petit code à barres attaché au composant, est associé à cet emplacement. Ce code à barres est alors demandé partout où l'emplacement est défini comme condition de départ.

Comme nous l'avons mentionné plus haut, les objets de produit sont reliés ensemble dans les arbres pour représenter la structure "comme-construite" du détecteur. Après l'exemple d'un sous-module, la figure 19 montre l'arborescence d'objets de produits résultant de l'affectation d'une structure alvéolaire et de 3 sous-unités à un sous-module.

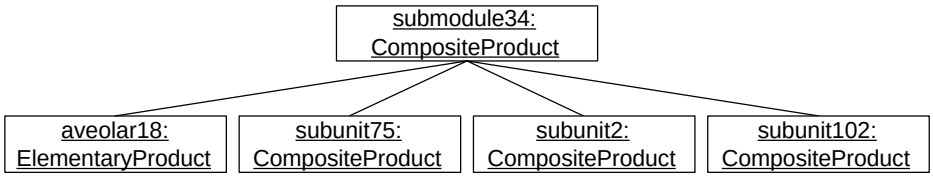



**Figure 19 Exemple d'arbre d'objets de produit**

Notez que les objets de produit représentant les sous-unités sont des exemples de la classe « Composite Product » car chaque sous-unité se compose d'une capsule et d'un cristal.

### 6.4.2 Objets d'activité et de définition d'activité

La figure 20 montre les définitions d'activité et des classes d'activité.

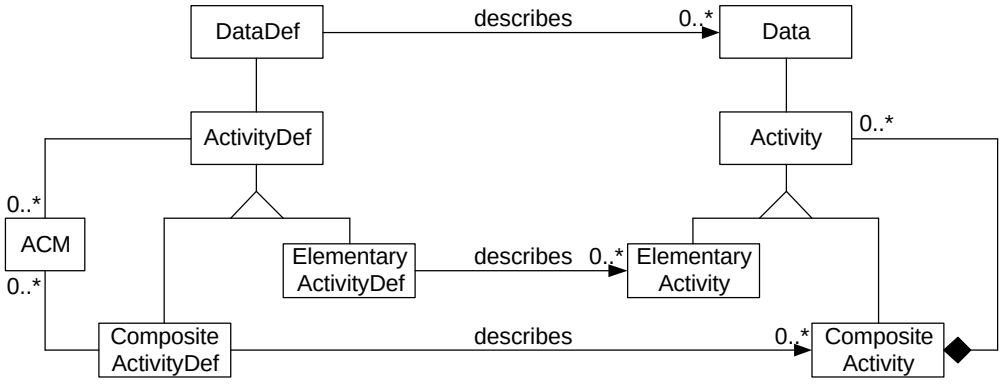



**Figure 20 Classes de définition d'activité**

Chaque objet de définition d'activité représente un type d'activité sur un graphe de définition de processus comprenant le nom, les instructions d'utilisateur, si on peut répéter ou sauter l'activité, et un lien à un objet de définition de données responsable de la définition du format de données des résultats d'activité. Chaque objet d'activité représente l'état d'une activité dans le cycle de construction d'un composant individuel du détecteur. Un objet d'activité a un lien à l'objet de données responsable d'enregistrer la valeur des résultats d'activité. Des objets de définition d'activité sont organisés en graphes acycliques orientés pour représenter des

définitions de processus, et des objets d'activité sont organisés en arbres pour représenter la position actuelle d'un composant du détecteur dans son cycle de construction.

Les objets composés et élémentaires de définition d'activité représentent les types composés et élémentaires d'activité respectivement. Les objets composés de définition d'activité sont reliés à leurs membres par les objets du Membre de Composition d'Activité (ACM) - un objet ACM par membre. Les objets ACM sont semblables aux objets PCM, sauf qu'ils représentent la composition d'activité par opposition à la composition de produit. Les objets ACM enregistrent actuellement les coordonnées graphiques de chaque membre, de sorte que des processus de construction peuvent être affichés aux utilisateurs dans un format graphique facile à comprendre. Certains ACM ne relient pas les objets composés de définition d'activité à leurs membres. Au lieu de cela ils représentent les divisions et les jonctions de la représentation de processus. Considérez la définition suivante de processus.

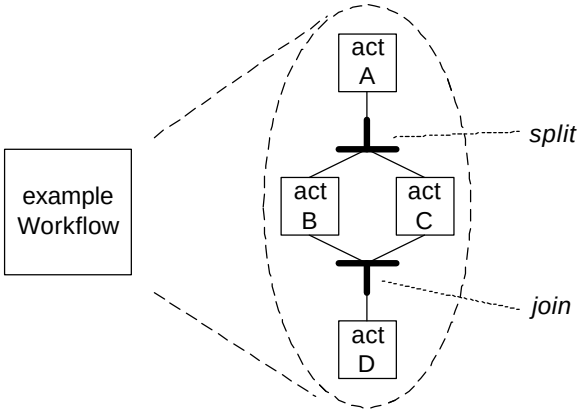



**Figure 21 Exemple de processus pour expliquer l'utilisation d'ACM**

La définition de processus a cinq types d'activité, un composé "example Workflow" formé de quatre processus élémentaires - A à D. La figure 22 montre les objets de définition d'activité employés pour représenter la définition ci-dessus de processus.

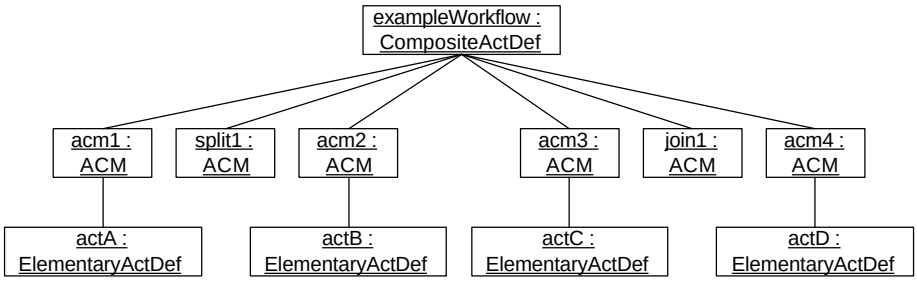



**Figure 22 Objets de définition d'activité représentant une définition de processus**

A part les coordonnées graphiques, un objet ACM enregistre également les liens qui relient le membre à d'autres membres dans la définition de processus. Par exemple "acm2", représentant l'activité B dans la définition ci-dessus de processus, contiendra les liens "split1" et "join1".

Comme mentionné auparavant, un arbre d'objets d'activité est employé pour représenter l'état actuel de processus d'un composant individuel du détecteur. La figure 23 est un diagramme d'objet montrant l'état de processus d'un composant du détecteur après l'exemple de définition de processus.

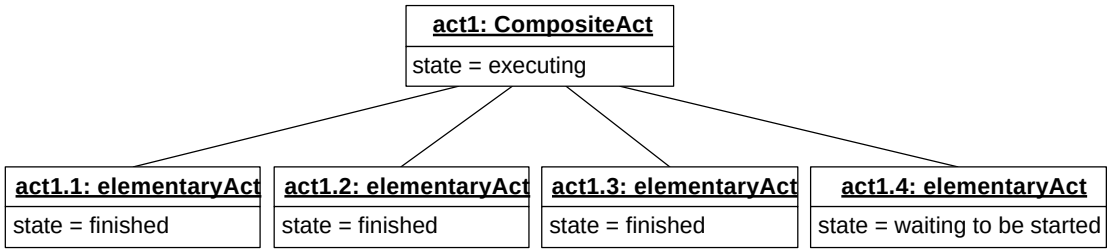



**Figure 23 Objets activités représentant un état du processus**

L'objet d'activité "act1" est décrit par le type d'activité "example Workflow" et les objets d'activité "act1.1" à "act1.4" par les types d'activité "A" à "D" respectivement. Le diagramme montre que les activités des types "A", "B", et "C" ont été terminées et que maintenant l'activité du type "D" attend pour être démarrée.

### 6.4.3 Objets de données et objets de définition de données

CRISTAL accepte trois types de base de données de résultats d'activité: champs, n-tuples et enregistrements. Des champs sont employés pour enregistrer les valeurs simples, par exemple une température en degrés Celsius. des n-tuples sont employés pour enregistrer des tables de longueur variable, où chaque colonne est une liste de champs du même type. Par exemple, un n-tuple peut représenter une table de deux colonnes donnant des temps en secondes dans la première colonne et des températures en degrés Celsius dans la seconde. Les enregistrements groupent des données ensemble dans des collections logiques. Les enregistrements peuvent contenir des champs, des n-tuples et d'autres enregistrements. La figure 25 est un diagramme de classe montrant les classes de définition de données et les classes de données.

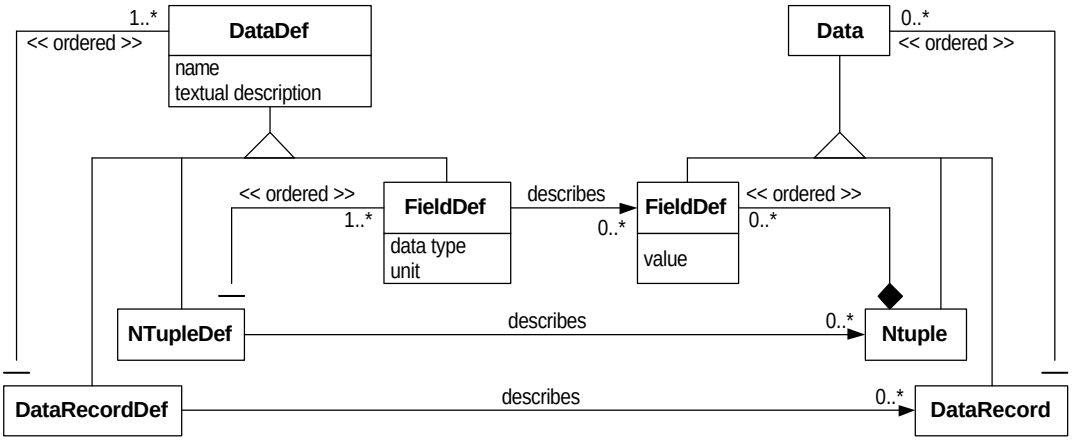



**Figure 24 Classes de données et de définition de données**

Les objets de définition de données représentent des variations utilisateur des types de données de base: enregistrement de champ, de n-tuple et. Les objets de données contiennent les valeurs des résultats. Les objets de définition de données sont arrangés dans les graphes acycliques orientés pour représenter des formats de données composites définis par l'utilisateur, et les objets de données sont arrangés dans des arbres pour représenter des résultats avec des données composites.

Un objet de définition de données permet à l'utilisateur d'indiquer le nom, le type et l'unité d'une valeur simple de données. Un objet de définition de n-tuple permet à l'utilisateur d'indiquer une table de longueur variable. L'utilisateur donne une liste d'objets de définition de champ représentant les colonnes des tables. Un objet de définition d'enregistrement permet à

l'utilisateur de créer les types composés de données en groupant ensemble d'autres objets de définition de données. Considérez l'écran de saisie suivant.

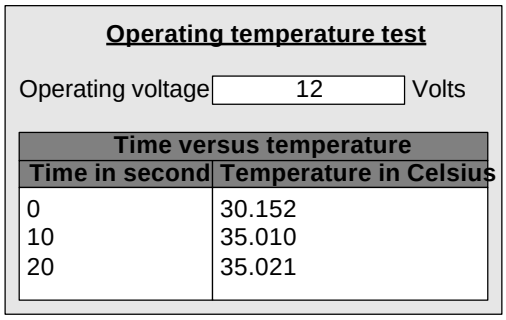



**Figure 25 Exemple d'écran de saisie de données**

Le format de données de l'écran de saisie montré ci-dessous en terme d'objet de définition de données.

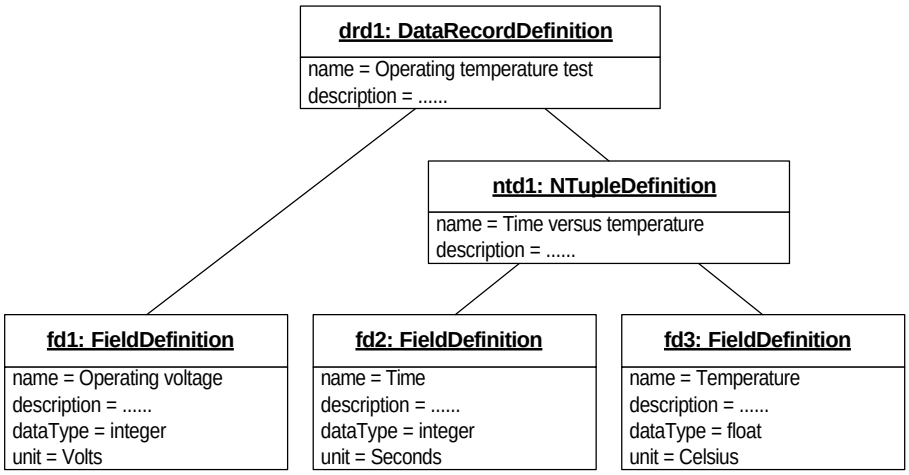



**Figure 26 Exemple d'objet de définition de données**

Le résultat de l'écran de saisie est montré ci-dessous en terme *d'objets de données.*

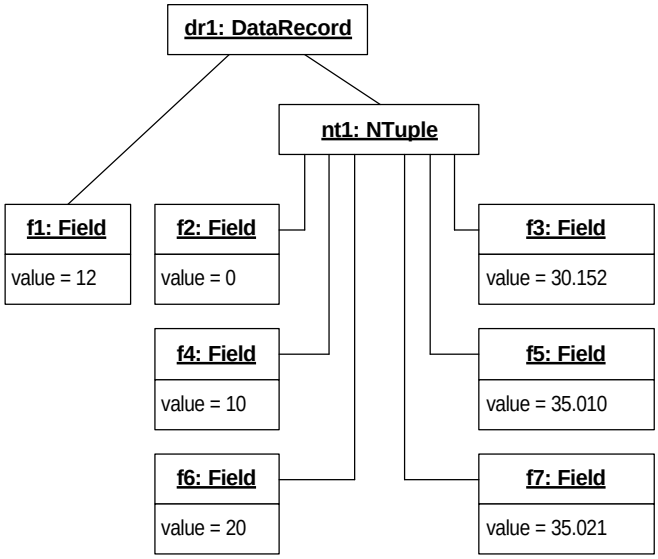



**Figure 27 Exemple d'objets de données**

## 6.5 Intégration de la composition du détecteur et du processus de production

Une partie importante du travail effectué par le projet CRISTAL était l'intégration de la composition du détecteur avec les processus employés pour contrôler la construction. Voir [Kovacs, 1999] pour une explication détaillée. Le modèle de données de CRISTAL montre ce qui est spécifique à la composition du détecteur, au processus de construction du détecteur, et ce qui colle les deux ensemble. La figure 28 montre que les graphes acycliques orientés des objets de définition de produits et de définition d'activité sont reliés ensemble par des objets de conditions de production.

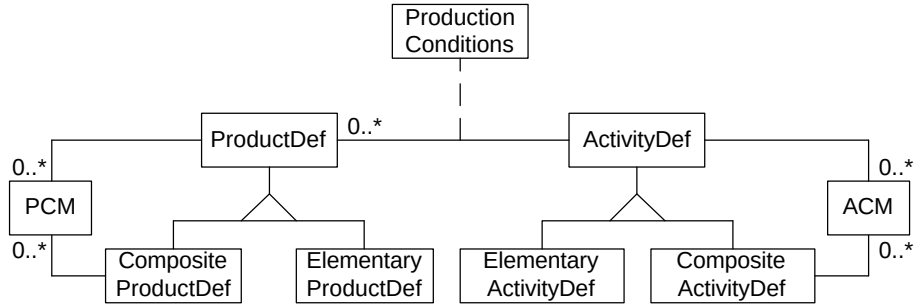



**Figure 28 Intégration de la composition du détecteur et du processus de production**

Chaque type de composant du détecteur a son processus de production. Le rapport « multiple à unique » entre la définition de produits et les classes de définition d'activité montre que beaucoup de types de composant du détecteur peuvent partager le même processus de production. Par exemple, les cristaux du ECAL dont les types diffèrent seulement par les dimensions des cristaux. Les conditions de production fournissent toute l'information spécifique sur le lien. Ceci inclut par exemple, début d'activité et conditions de fin. Dans le cas des conditions initiales, un objet de conditions de production peut contenir une table reliant les conditions de début aux activités, c'est à dire les objets de PCM aux objets d'ACM.

Le modèle intégré permet à des processus d'être réutilisés. Par exemple, considérons les cristaux de tungstate de plomb du ECAL. Ils appartiennent tous à la même famille de produit, c'est à dire le cristal, mais sont subdivisés en beaucoup de types, où chaque type représente une forme différente de cristal. Cependant, quelle que soit la forme d'un cristal, le même processus s'applique. Par conséquent en utilisant le modèle ci-dessus, tous les différents types de cristal du ECAL peuvent partager la même définition de processus, les conditions de production indiquant les conditions de fin d'activité spécifiques à chaque forme de cristal, c'est à dire les conditions de fin à réunir par les résultats des mesures 3D qui dépendent de la forme du cristal.

## 6.6 Créer des versions de descriptions d'éléments

Un arrangement de production décrit la composition "à-construire" du détecteur ainsi que les définitions de processus exigées pour contrôler sa construction. Le schéma de production est créé dans le système central, puis distribué aux centres de fabrication. C'est une des méthodes employées pour contrôler la construction distribuée du détecteur.

Le schéma de production est la somme de tous les objets de description d'élément dans le modèle de données de CRISTAL. L'évolution des processus de conception et de construction

du détecteur est reflétée dans de nouvelles versions de la définition de schéma de production. Les étapes pour produire la première version de l'arrangement de production sont les suivantes:

1. Créer le premier ensemble d'objets de description d'élément qui indiquent la composition "à-construire" du détecteur et des processus de production qui contrôleront sa construction.
2. Modifier les différents objets de description d'élément jusqu'à ce qu'ils soient prêts pour la production. Les nombres de version de différents objets de description d'élément sont incrémentés pendant qu'ils sont modifiés.
3. Quand toutes les modifications nécessaires ont été faites, contrôler la cohérence globale des objets de description d'élément, c'est à dire la cohérence du schéma de production.
4. Si le schéma de production est logique, alors le transmettre aux centres de fabrication de sorte qu'ils puissent commencer la production.

Juste avant que le schéma de production soit envoyé aux centres de fabrication dans l'étape 4, le système de CRISTAL lui donne un numéro de version, qui dans ce cas-ci sera 1. Le réglage de la version de schéma de production implique de passer le numéro de version vers chacun des objets de description d'élément, qui en retour se rappellent leur numéro de version actuelle comme celui qui est valide pour la version courante de schéma de production. En effet un instantané des présentes versions des objets de description d'élément est pris et est attribué comme version du schéma de production. Créer de nouveaux objet de description d'élément, effacer ou modifier les anciens, puis fournir la nouvelle version aux centres de fabrication crée de nouvelles versions de la définition du schéma de production.

La figure 29 résume le contenu d'un objet de description d'élément et d'un objet d'élément dans le contexte du mécanisme de version. Notez que "PS" signifie « schéma de production ».

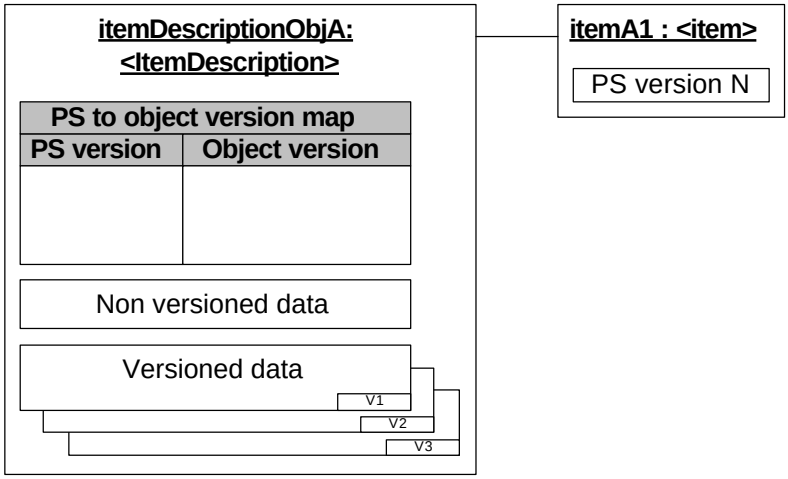



**Figure 29 Mécanisme de version de CRISTAL**

Les données enregistrées dans un objet de description d'élément sont divisées en 2 sections, ces données qui peuvent avoir une version et ceux qui ne le peuvent pas. Les objets d'élément connaissent seulement la version de schéma de production qu'ils suivent, ils ne connaissent pas les versions associées d'objet de description d'élément. Afin qu'un objet d'élément ait la description correcte dans la version correcte, il doit y avoir un lien à son objet de description d'élément correspondant ainsi que le numéro de version du schéma de production qu'il utilise. Quand un composant du détecteur est enregistré dans le système CRISTAL, il suit la dernière version du schéma de production, c'est à dire les objets d'élément qui le représentent utilisent

la dernière version de l'objet de description d'élément correspondant. Le mécanisme de version permet à des objets d'élément de changer la version de l'objet de description d'élément qu'ils utilisent. Ceci ouvre la porte à la possibilité de permettre à chaque composant individuel du détecteur d'essayer d'utiliser la dernière version du schéma de production aussitôt que possible. En particulier la capacité de chaque composant du détecteur de passer à la dernière version de processus est d'un grand intérêt. La politique actuelle d'évolution de processus de CRISTAL est la suivante:

Quand le coordinateur crée une nouvelle version d'une définition de processus, le gestionnaire de définition de schéma de production divise la version précédente en deux sections. La première section commence au début de la définition de processus et se termine à la première modification sans l'inclure. La deuxième section contient la modification plus le reste de la définition de processus. La deuxième section est désignée sous le nom de "zone de changement". Quand la nouvelle version de la définition de processus arrive à un centre de fabrication, les produits suivant la version précédente se déplaceront à la nouvelle version s' ils n'ont pas commencé une activité dans la « zone de changement ». Une solution possible, mais pas encore testée pour l'évolution de processus est que, lorsque des utilisateurs définissent une nouvelle version de processus, ils définissent également comment migrer depuis chaque étape de la version précédente.

## *6.7 Conclusion*

Le modèle de données de CRISTAL prouve qu'une approche orientée description est un moyen approprié de modéliser la composition d'un grand détecteur HEP et des processus qui commandent et contrôlent sa construction. CRISTAL modélise les éléments et leurs descriptions selon deux ensembles d'objets, leur donnant la capacité d'évoluer indépendamment les uns des autres. En mettant l'information de description d'élément dans son propre objet, Le schéma de base de données aura moins de chance de nécessiter des évolutions. Il est maintenant important de donner à l'utilisateur une interface flexible et facile à utiliser qui cache les complexités du modèle de données de CRISTAL. Les opérateurs construisant le détecteur sont plus intéressés à exécuter des tâches et à assembler des composants, que par le modèle de données qui stocke l'information résultante.

# 7 Développement de l'interface utilisateur de l'Atelier

## *7.1 Introduction*

Le deuxième objectif de cette thèse est de visualiser le processus d'assemblage et de caractérisation du détecteur. En particulier ce travail avait la responsabilité de réaliser l'interface utilisateur de l'Atelier d'un centre de fabrication, interface désignée dorénavant sous le nom d'interface opérateur. Ce chapitre commence par décrire les contraintes imposées à un réalisateur de GUI dans l'environnement de CRISTAL. Ceci est suivi d'une étude des techniques existantes de développement d'interface utilisateur. La technique appropriée est alors choisie et développée. Le développement des fonctionnalités de processus de l'interface opérateur utilisant les techniques adoptées est alors présenté. Le chapitre conclut avec une discussion de la façon dont les techniques peuvent être appliquées au développement de GUI en général.

## *7.2 Contraintes imposées au développement d'interfaces (GUI) dans CRISTAL*

Le facteur principal imposé à un réalisateur d'interface utilisateur dans l'environnement de CRISTAL est la stratégie de développement de prototypage rapide. La couche de logique d'entreprise de CRISTAL changera rarement car elle est conçue pour être aussi générale que possible, toutefois l'interface utilisateur graphique (GUI) devra passer par beaucoup d'itérations en raison du caractère très détaillé de ses fonctionnalités. Vu les contraintes sévères de temps, il a fallu trouver une technique de développement qui permettrait à des interfaces utilisateur d'être développées, et d'évoluer avec un minimum de modifications au code existant.

## *7.3 Techniques de développement disponibles*

Les interfaces utilisateur de CRISTAL sont développées en Java. Au moment où la présente partie des travaux de thèse était menée à bien, les moyens existant pour le développement d'interfaces utilisateur étaient comme suit:

1. Écrire le code en Java en utilisant seulement un éditeur de texte.
2. Utiliser un générateur graphique de GUI.

Ecrire du code Java à la main n'est pas aussi difficile qu'il peut paraître. Les bibliothèques de classe de Java sont très puissantes et une fenêtre peut être assemblée en quelques heures. Lorsque ce travail de thèse commençait, les générateurs d'interface économisaient le temps en permettant à des développeurs de dessiner l'organisation de leurs fenêtres et de leurs écrans. Cependant il était difficile de travailler sur le code produit par ces générateurs. Par comparaison avec le développement uniquement avec un éditeur de texte, le temps gagné sur le dessin de l'interface était reperdu dans l'effort d'ajouter la logique des événements au milieu de ce code difficile à comprendre. Cette thèse a donc choisi d'utiliser un éditeur de texte sans l'aide d'un générateur de GUI. Il faut préciser cependant que les générateurs de GUI ont évolué au cours des 3 années de cette thèse, et sont maintenant assez mûrs pour fournir une solution viable. Si ce travail de thèse était commencé maintenant, un constructeur de GUI serait utilisé. Le développement d'une interface utilisateur utilisant seulement un éditeur de texte a exigé une technique de développement pour s'assurer que le code sera évolutif.

## *7.4 Techniques pour permettre l'évolution des interfaces*

Pendant la phase de test de l'interface opérateur, les utilisateurs ont changé leurs conditions environ une fois par semaine. Par conséquent la conception du code a eu besoin d'une certaine qualité de souplesse du genre « insérez et démarrez » pour permettre à des composants de GUI d'être ajoutés et retirés avec un minimum de changements au code existant.

Une stratégie possible de conception serait de distribuer le code gérant le comportement dynamique de la fenêtre au-dessus de ses composants de GUI. Chaque composant serait responsable de mettre à jour les autres composants quand il reçoit la commande utilisateur appropriée. Par exemple, une liste graphique pourrait être responsable d'activer ou d'invalider un bouton selon qu'un quelconque de ses éléments est choisi. Cette stratégie de conception produit une solution évolutive. Si un utilisateur demande qu'un composant du GUI soit retiré de la fenêtre, alors le développeur doit rechercher dans le code source tous les composants du GUI pour supprimer toutes les parties de code s'adressant au composant à retirer.

### 7.4.1 Utilisation du modèle d'observateur et de médiateur

Au lieu d'essayer de trouver une solution maison, on a décidé de chercher les modèles de conception publiés. Les modèles pour le développement de logiciel sont des relations documentées des meilleures pratiques pour la solution de problèmes qui se reproduisent dans le développement de logiciel. Le but de ces modèles est de réutiliser la connaissance et le code, évitant de réinventer la roue. Pour une explication sur la Toile des concepts essentiels et de la terminologie voir [Appleton 2000]. Le modèle de médiateur décrit dans [Gamma et al, 1995] était la solution choisie parce que le comportement exact des composants de GUI était connu, mais leurs interdépendances étaient non structurées.

Pour résumer, le modèle de médiateur se compose de beaucoup d'objets cousins (ou « collègues ») qui ne se connaissent pas. Le seul objet qu'ils connaissent est un médiateur unique. Quand un objet cousin change d'état il informe le médiateur. En réponse le médiateur calcule les conséquences pour les autres objets cousins et leur envoie les messages nécessaires "changez d'état". La figure 30 est un diagramme d'échanges donnant un exemple de scénario.

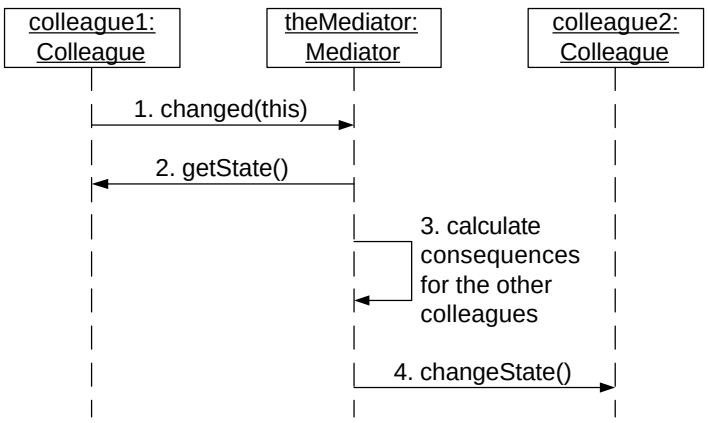



**Figure 30 Exemple de scénario dans un modèle de médiateur**

On a décidé de centraliser le comportement dynamique de l'affichage du processus dans un objet unique de médiateur. On a également décidé d'employer le modèle d'observateur [Gamma et al, 1995] pour coupler faiblement les composants du GUI au médiateur. Pour

récapituler, le modèle d'observateur a un objet unique Sujet, observé par beaucoup d'objets observateur. Le sujet a un ensemble de références pointant sur ses observateurs et mis à jour par deux méthodes fournies par le sujet: attach() et detach(). Ceci permet d'ajouter des observateurs au Sujet pendant l'exécution, éliminant la nécessité de modifier le code du Sujet.

Le modèle d'observateur est utilisé quand un changement dans un objet exige une notification à d'autres objets non connus au temps de sa création. L'objet dont le changement d'état doit être signalé s'appelle « Sujet », et les objets qui doivent être prévenus s'appellent les "observateurs". La figure 31 est un diagramme d'ordre donnant un exemple de scénario.

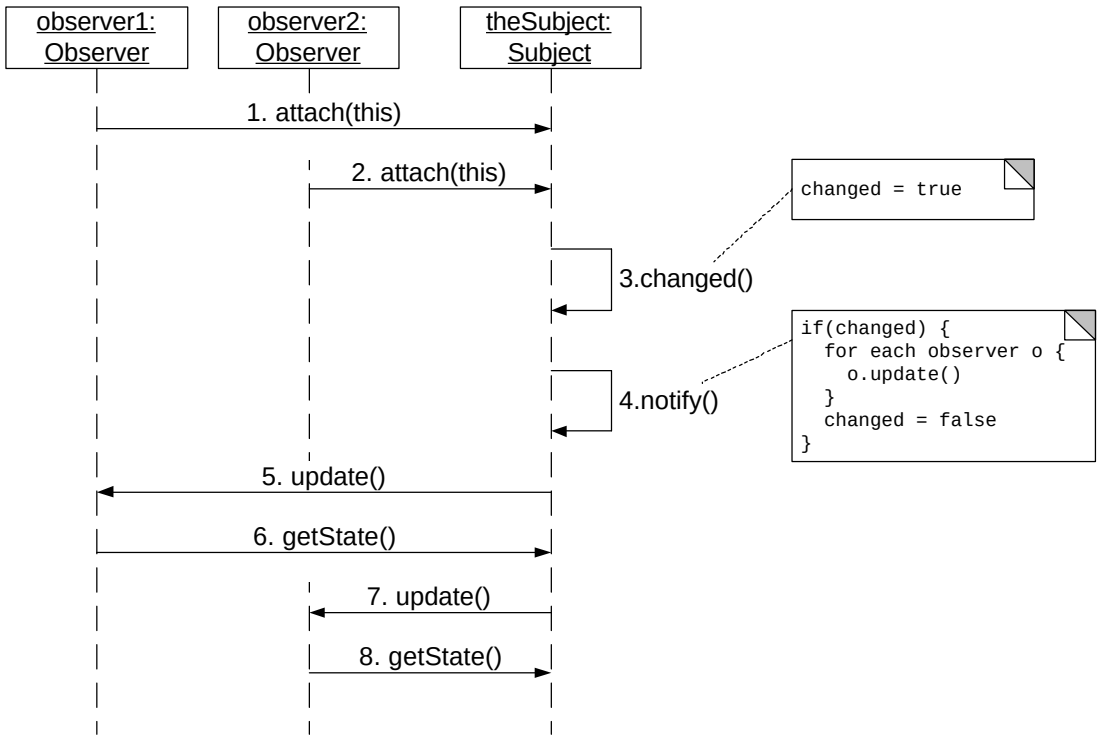



**Figure 31 Exemple de scénario de modèle d'observateur**

Le sujet a un ensemble de références pointant sur ses observateurs. Dans les étapes 1 et 2 du scénario ci-dessus, deux observateurs s'ajoutent à cette liste en appelant la méthode attach() du sujet. A l'étape 3 le sujet a changé l'état et appelle donc sa méthode interne changed(), qui place un indicateur interne pour indiquer le changement d'état. La méthode changed() donne au sujet la capacité d'informer ses observateurs indépendamment du moment où il change d'état, par exemple il peut informer ses observateurs toutes les dix secondes. A l'étape 4 le sujet a décidé d'informer ses observateurs de tous ses changements d'état, en appelant sa méthode interne notify(). Cette méthode vérifie si l'état du sujet a changé, et si oui, appelle la méthode update() de chaque observateur (étapes 5 et 7). En réponse à l'appel de leur méthode update(), les observateurs demandent l'état du sujet en appelant sa méthode getState() dans les étapes 6 et 8. Les observateurs peuvent également se retirer de la liste à l'intérieur du sujet. Ceci n'est pas montré dans le scénario ci-dessus, mais ils le font en appelant la méthode detach() du sujet. Un des avantages principaux d'utiliser le modèle d'observateur, est que des observateurs sont ajoutés à et retirés du sujet en cours d'exécution sans aucune modification au code.

### *7.5 Estimer les implications sans connaître les objets « collègues »*

Le médiateur doit pouvoir calculer les conséquences du changement d'état d'un collègue sans connaître directement chaque collègue individuel. Autrement il est impossible de coupler faiblement le médiateur et ses collègues. Pour faire ceci, le médiateur doit savoir les dépendances entre les différentes données d'éléments affichées et manipulées par la fenêtre. Les composants du GUI indiquent au médiateur de changer l'état de ces données d'éléments. Dans sa réponse, le médiateur met à jour les dépendances des données d'éléments puis informe tous les composants du GUI des changements en leur envoyant un masque qui identifie quelles données d'éléments ont changé d'état. La figure 32 aide à expliquer ceci.

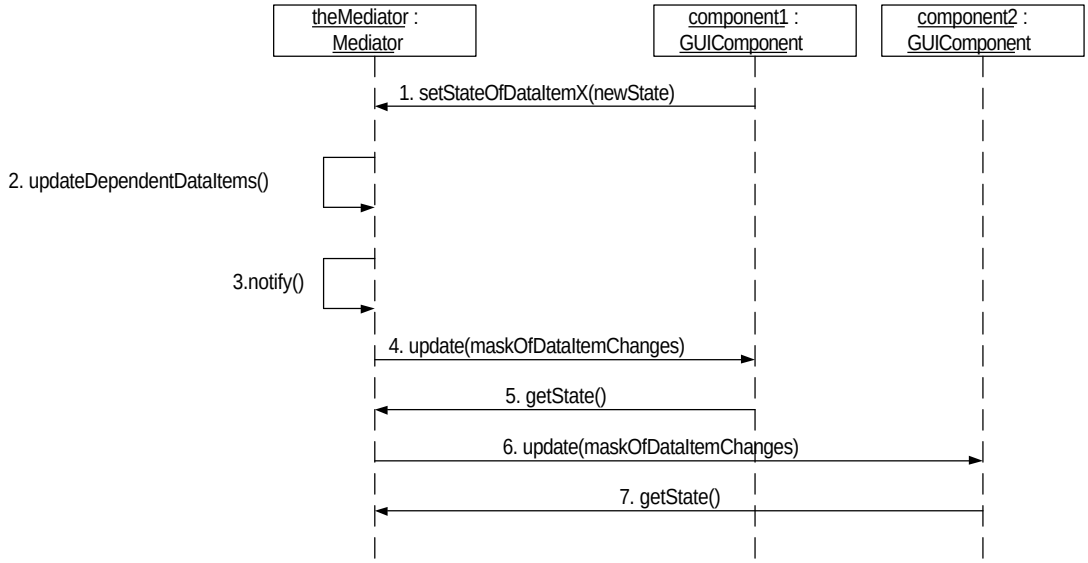



**Figure 32 Utilisation du modèle d'observateur avec le modèle de médiateur**

Il y a deux composants du GUI dans le scénario ci-dessus, étiquetés 1 et 2. Le composant 1 du GUI demande au médiateur d'affichage de processus de placer l'état de la donnée d'élément X à "newState". Le médiateur met à jour les données d'éléments dépendantes de la donnée d'élément X, et informe alors les composants du GUI du changement. Chaque composant du GUI lit le masque et demande au médiateur l'état des données d'éléments qu'ils représentent.

## *7.6 Fonctionnalités de processus de l'interface opérateur*

Avant d'expliquer comment les techniques développées dans la section précédente ont été employées pour développer la fonctionnalité de processus de l'interface d'opérateur, il est nécessaire de donner une description rapide de ces fonctionnalités.

L'interface opérateur est grande, complexe et multi aspects. Les opérateurs et les superviseurs de centre l'emploient pour exécuter des activités de production et pour assembler des composants du détecteur. Les superviseurs de centre peuvent faire tout ce que font les opérateurs, plus quelques fonctions privilégiées. L'interface est orientée produit, ce qui veut dire qu'un utilisateur indique avec quel composant individuel du détecteur il souhaite travailler, et puis exécute les activités de construction et d'assemblage sur ce composant. La figure 33 est un organigramme d'interface montrant les fenêtres d'interface et les chemins les reliant.

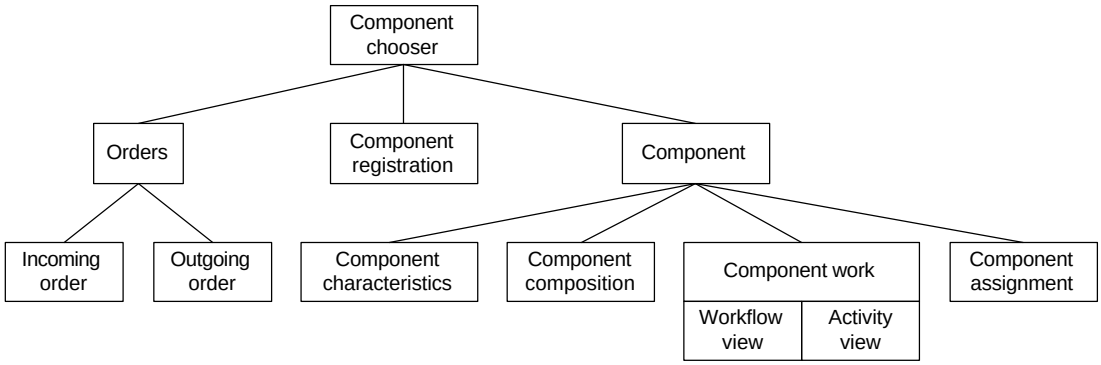



**Figure 33 Organigramme d'interface opérateur**

La fenêtre de « choix de composant » permet à l'utilisateur de dire à l'interface sur quel composant il désire travailler. La fenêtre de commandes et ses fenêtres filles permettent à des utilisateurs de recevoir et d'envoyer des commandes. Notez que les commandes elles-mêmes sont créées dans le système central, car ceci est une forme de commande centralisée. Les utilisateurs donnent le seul identificateur (code à barres) et le type de chaque nouveau composant du détecteur dans la fenêtre d'enregistrement de composant. La fenêtre composant apparaît une fois qu'un composant du détecteur a été choisi pour le travail. De cette fenêtre, les utilisateurs peuvent aller aux fenêtres filles pour visualiser les caractéristiques des composants (fenêtre de caractéristiques de composants), pour gérer la composition des composants (les fenêtres de composition et d'affectation de composants), et pour exécuter des activités de processus (fenêtre de travail du composant). La fenêtre de travail de composant, à la différence des autres, est divisée en deux vues: "processus" et "activité". La vue de processus montre le cycle de construction du composant du détecteur sur lequel on travaille. Avec cette vue les utilisateurs peuvent obtenir la liste de prochaines activités possibles et exécuter des fonctions de gestion de processus. Quand un utilisateur choisit de travailler sur une activité la fenêtre commute sur la vue d'activité où l'utilisateur peut voir les instructions d'activité, les conditions de début et la forme de saisie de résultats. Cette vue permet à des utilisateurs d'exécuter des activités.

CRISTAL attribue à chaque composant individuel du détecteur son propre cycle de construction qui suit une définition de processus indiquée pour son type. Cette section discute comment les fonctionnalités de processus de l'interface d'opérateur permettent à des utilisateurs de contrôler et de suivre l'exécution de leurs activités de construction sur des composants du détecteur.

### 7.6.1 Fenêtre de choix du composant

La fenêtre de choix de composant est montrée sur La figure 34. C'est la première fenêtre présentée à l'utilisateur.

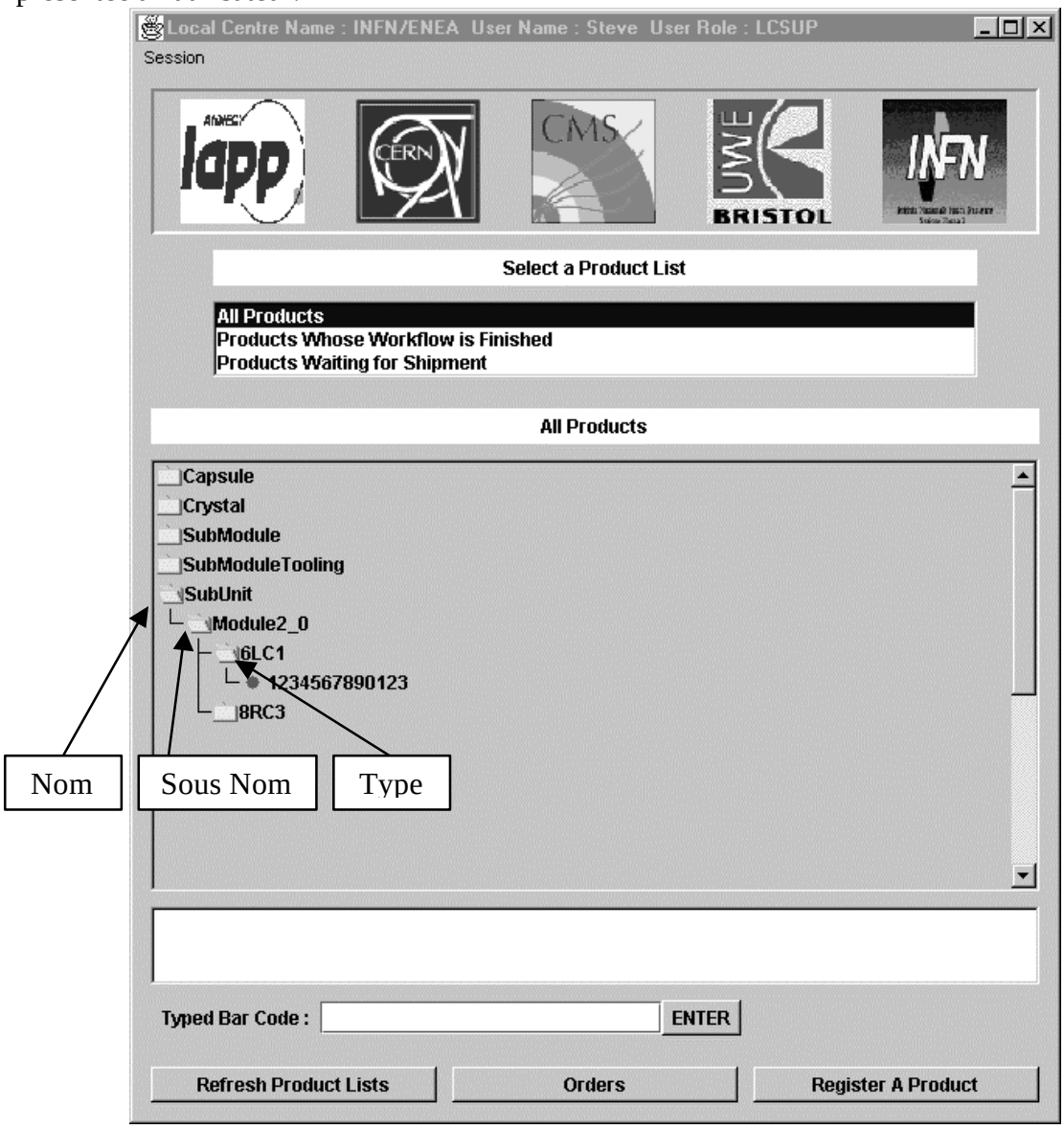



**Figure 34 Fenêtre de choix de produit**

Un composant du détecteur peut être choisi en utilisant trois méthodes différentes. Les utilisateurs peuvent lire un code à barres attaché au composant, le taper manuellement, ou ils peuvent choisir le composant à partir d'une liste. Il y a trois listes à partir desquelles un composant peut être choisi: tous les composants situés au centre, composants dont le processus de construction est terminé, et composants attendant l'expédition à un autre centre. Chaque liste est affichée comme liste d'arbres, semblable à celle d'un afficheur de fichier. La structure de chaque arbre est basée sur un 3-tuple: "nom", "sous-nom" et "type". le "nom" correspond normalement à une famille de composants. La photo de l'écran de la fenêtre de choix de composant montre les familles: capsule, cristal, sous-module, outils-sous-module, et sous-unité. "sous-nom" et "type" aident l'utilisateur à trouver le codes à barres du composant efficacement. Par exemple, dans l'écran ci-dessus il y a des composants du détecteur avec le 3-tuple "sous-unité - module2_0 - 6LC1". L'utilisateur a choisi ces derniers parce qu'il est

commode de grouper d'abord les sous-unités par leurs composés, et puis par leur type. Une fois que l'utilisateur a choisi un composant du détecteur, la fenêtre de composant correspondante est affichée.

### 7.6.2 Fenêtre de composant

La fenêtre de composant est montrée ci-dessous:

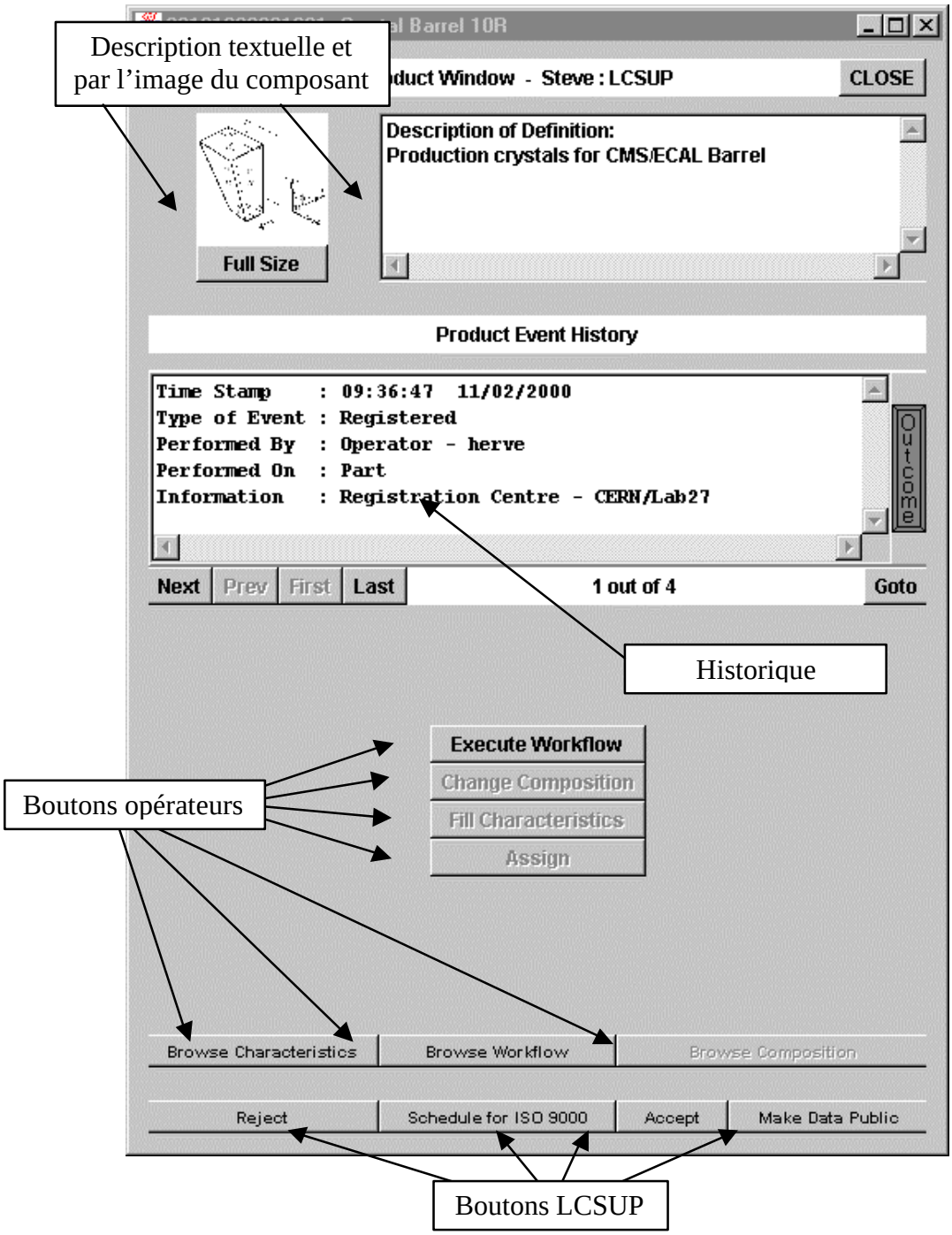



**Figure 35 Fenêtre de composants**

LE HAUT DE LA FENETRE montre une image et donne une description textuelle du composant du détecteur en cours de traitement. En dessous, l'histoire du composant du détecteur. Cette histoire ne montre pas quand on a exécuté des activités de construction sur le

composant, car ceci est la responsabilité de l'historique de processus. L'historique de composant enregistre les types suivants d'événement:

Composant enregistré
Composant rejeté
Fils alloué au composant
Fils retiré du composant
Expédition du Composant en cours
Composant reçu au centre de production
Composant alloué à un parent
Composant retiré d'un parent
Version de Composant corrigée
Données Composant mises à public
Données Composant mises à private

Au centre de la fenêtre et juste en-dessous sont les boutons disponibles pour l'opérateur. Ces boutons ouvrent les fenêtres : travail composant, composition du composant, caractéristiques du composant et affectation du composant. La ligne inférieure de boutons fournit à un SUPervisor de centre local (LCSUP) quatre actions privilégiées. Premièrement, il peut rejeter le composant à tout moment. Deuxièmement et troisièmement, pour un composant avec des résultats d'activité qui ne satisfont pas les conditions de fin, un LCSUP peut programmer ce composant pour une procédure de contrôle de qualité ISO9000, qui permet d'accepter le composant de nouveau dans la construction. Quatrièmement, un LCSUP peut mettre les droits d'accès des données de construction attachées à un composant du détecteur dans l'état privé ou public. Par défaut les droits d'accès sont privés, signifiant que toutes les données de construction rassemblées au sujet d'un composant sont réservées au centre(s) de fabrication qui travaille sur ce composant. Les droits d'accès publics signifient que tous les utilisateurs du système de CRISTAL peuvent voir les données de construction.

### 7.6.3 Vue « Processus » de la fenêtre de travail sur composant

La fenêtre de travail sur composant est consultée par l'intermédiaire de la fenêtre composant avec le bouton "exécuter processus". A la différence des autres fenêtres d'interface opérateur, la Fenêtre travail sur composant a deux vues: processus et activité. La vue processus affiche le graphe de définition de processus, l'historique du processus et la liste des prochaines activités possibles. Les utilisateurs peuvent exécuter des actions de gestion de processus et choisir leur prochaine activité. Les utilisateurs contrôlent le processus d'un composant en programmant des activités pour être sautées ou répétées, et en remettant à l'état initial des activités en marche de sorte qu'ils soient prêts à être recommencés.

La figure 36 est une vue d'écran montrant la vue processus de la Fenêtre Travail sur composant.

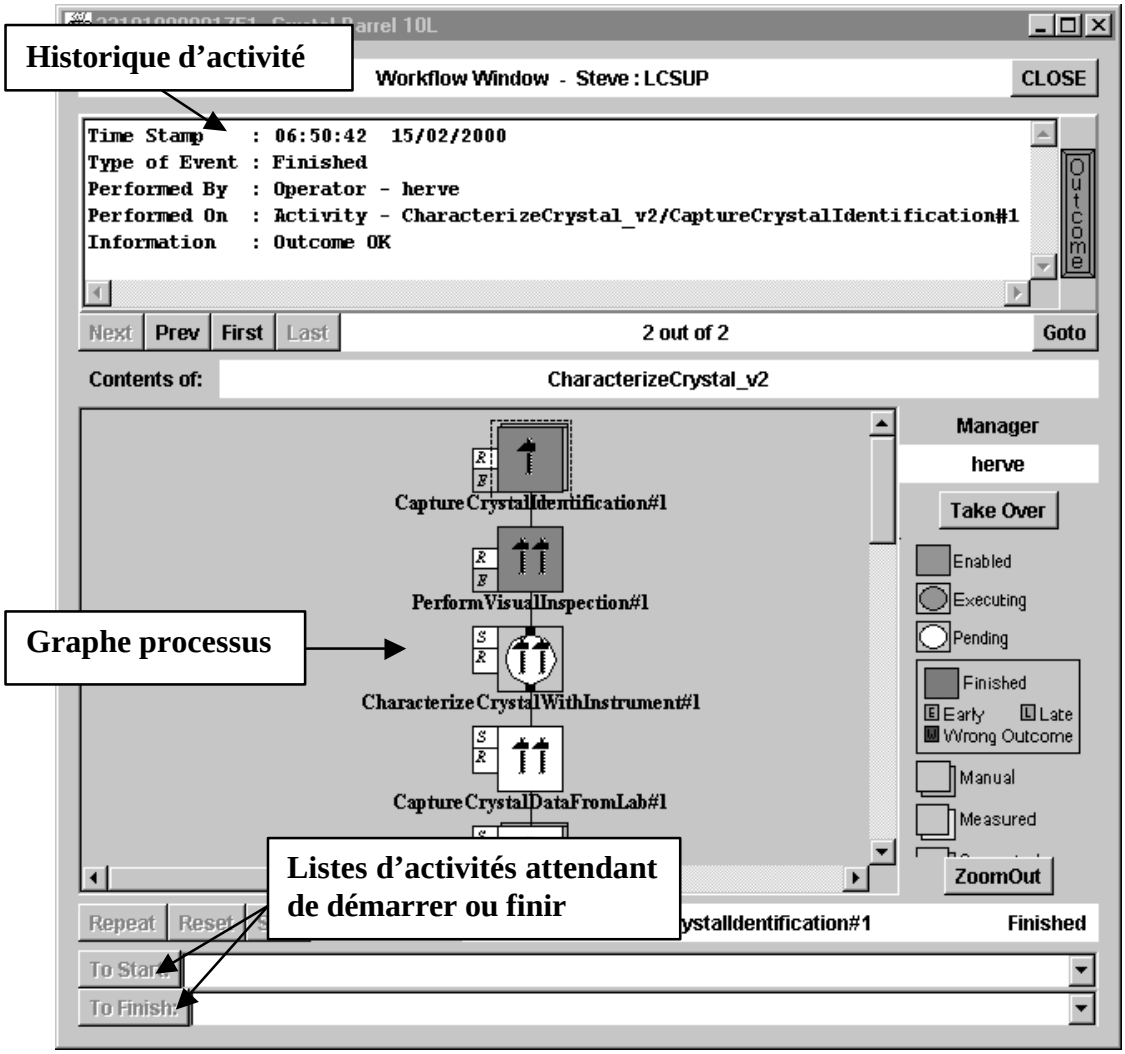



**Figure 36 Vue processus de la fenêtre « travail sur composant »**

Au milieu de la Fenêtre se trouve le graphe de processus montrant l'ordre partiel des activités qui peuvent être exécutées sur le composant du détecteur en cours. Le choix d'une activité dans le graphe avec un simple clic de souris affiche son histoire en haut de la Fenêtre. Le bas de la fenêtre a deux listes, activités attendant pour être commencées (des activités permises) et activités attendant pour être terminées (activités en cours d'exécution). Comme mentionné auparavant, chaque composant du détecteur a un objet correspondant du logiciel appelé un

gestionnaire de produit. Il y a un moteur de processus dans chaque gestionnaire de produit qui traverse le graphe du processus du composant en activant les nœuds au passage. L'ensemble des activités permises à un moment donné représente la liste de prochaines activités possibles.

Indépendamment des rôles d'exécution et de gestion des exploitants de centres de fabrication il y a également dans CRISTAL les rôles utilisateur d'opérateur et du superviseur de centre. Comme indiqué auparavant, un superviseur de centre peut faire tout qu'un opérateur peut faire plus quelques fonctions privilégiées supplémentaires. Dans le cas de la fenêtre de travail sur produit, chaque activité a un niveau de droits d'accès, qui peut être opérateur ou superviseur de centre. Un opérateur peut uniquement exécuter des activités avec les droits d'accès d'opérateur, tandis qu'un superviseur de centre peut exécuter celles-ci plus celles avec les droits du superviseur de centre.

Les historiques d'activité affichés en haut de la fenêtre montrent les types suivants d'événement.

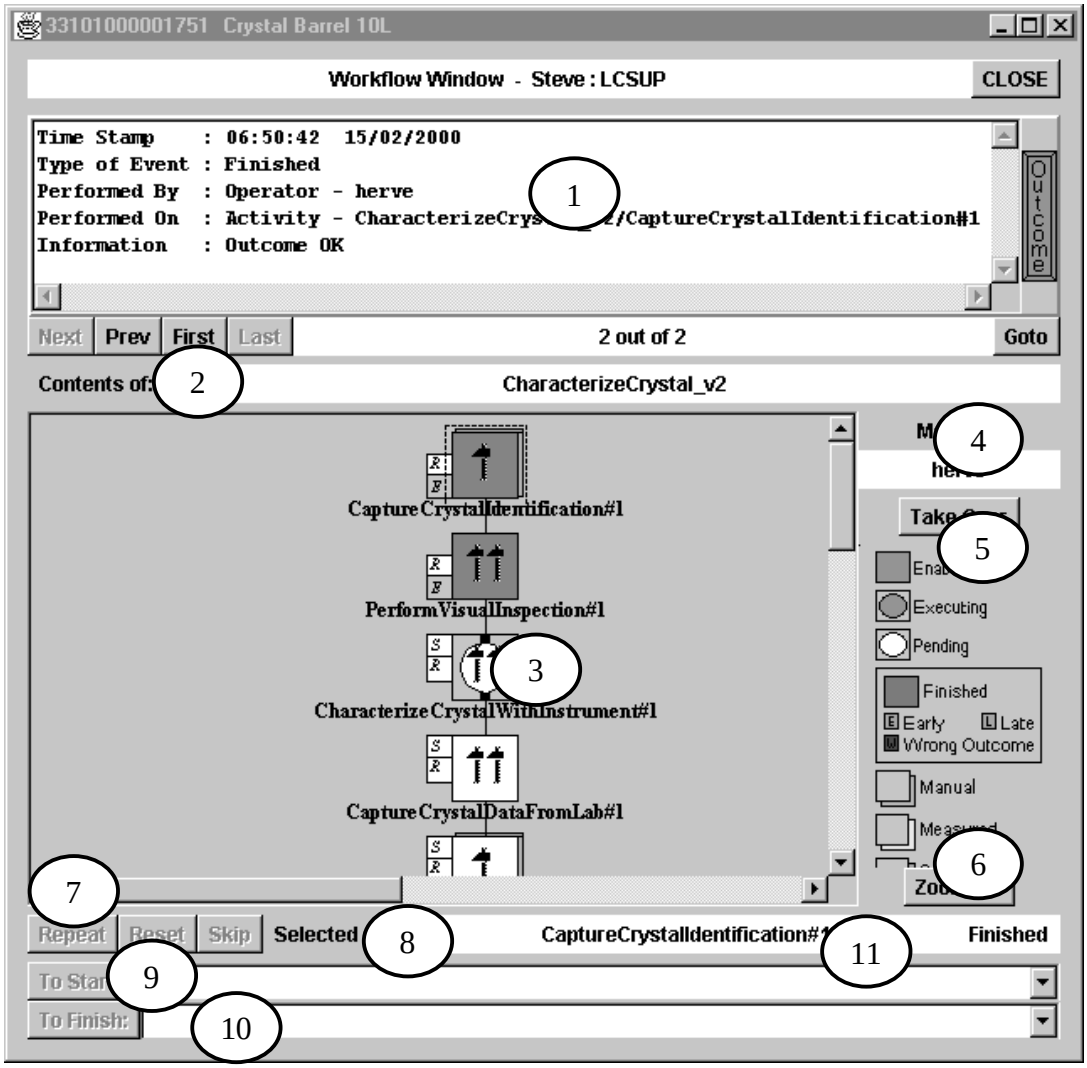
| | |
|---|---|
| Activité démarrée | Activité sautée |
| Activité finie | Activité hors délai |
| Activité ignorée | Sous-activité à répéter |
| Activité réinitialisée | Sous-activité à sauter |
| Activité reprise | |

Quelques événements ont des résultats, par exemple l'événement « activité terminée » Des résultats peuvent être visualisés en cliquant le bouton de résultats à la droite de l'affichage d'historique.

Un double clic sur une activité permise ou en exécution dans le graphe de processus, ou le choix d'activité à partir des listes d'activités "à commencer" et "à terminer", met la fenêtre de travail sur produit dans la vue d'activité. Cette vue montre les conditions de début, les instructions et le format de données des résultats de l'activité choisie. Les utilisateurs peuvent commencer, terminer ou ignorer l'activité avec cette vue.

## *7.7 Développement des fonctionnalités « processus » de l'interface opérateur*

Le défi principal en développant l'interface opérateur était comment traiter la complexité de la vue de processus de la fenêtre de travail sur produit. La figure 37 montre les composants du GUI qui forment la vue processus:




1. SelectedActHistoryPanel
2. ContentsOfPanel
3. WfPanel
4. CompositeActManagerNamePanel
5. TakeOverCompositeActPanel
6. ZoomOutWfPanel
7. ManagementButtonsPanel
8. SelectedActPanel
9. ActsToStartPanel
10. ActsToFinishPanel
11. SelectedActStatePanel

**Figure 37 Composants GUI de la vue « processus »**

Après les techniques présentées dans ce chapitre, un graphe de dépendance des données a été produit pour modéliser les dépendances entre les données d'éléments affichées et manipulées par la vue processus. Le résultat est montré sur la figure 38.

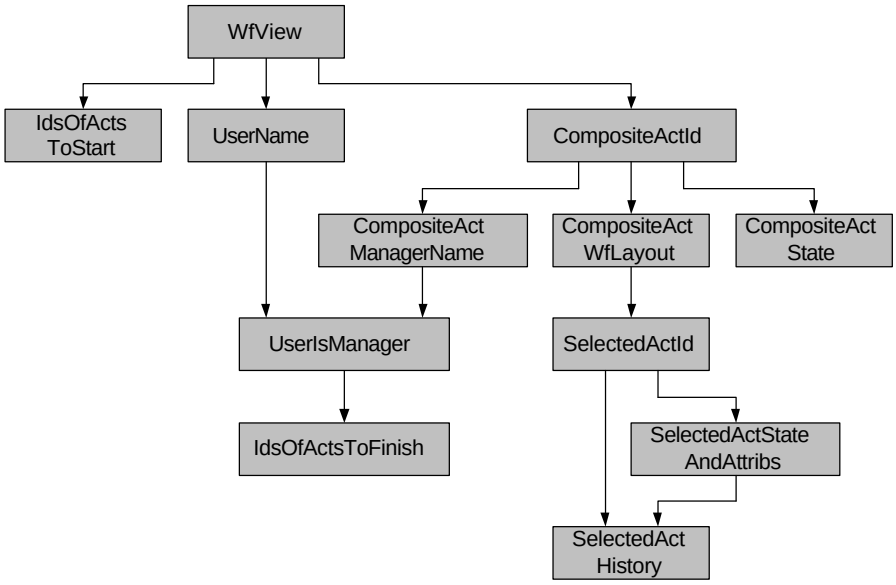



**Figure 38 Graphe de dépendances entre les données d'éléments de la vue processus**

Chaque nœud du graphique représente une donnée élémentaire affichée et/ou manipulée par la vue processus. Un arc reliant une donnée d'élément A à une autre donnée d'élément B, de direction A-B, indique qu'un changement de l'état de A aura comme conséquence un changement de l'état de B. Par exemple le graphe montre que changer l'activité sélectionnée (les "selectedActId") entraîne la mise à jour de l'état, des attributs et de l'historique de l'activité choisie.

## *7.8 Conclusion*

En conclusion, le développeur d'un GUI complexe peut produire un code facilement évolutif et extensible en utilisant les modèles de médiateur et d'observateur. Pour combiner le modèle d'observateur avec le modèle de médiateur, il est nécessaire de développer la logique de l'objet de médiateur de telle manière qu'il n'ait pas besoin de connaître chaque composant de GUI directement. La solution prise pour la vue processus de la fenêtre de produit était de modéliser les dépendances entre les données montrées et/ou manipulées.

# 8 Détermination de la capacité de production

## *8.1 Introduction*

Le troisième et objectif final de cette thèse est de développer des techniques pour mesurer la capacité des ressources de production pendant la construction du détecteur. Pendant la pleine phase de construction il est important de pouvoir déterminer la capacité de production pour répondre à la question de savoir si les dates limites de construction seront respectées. Le système de Planification des Moyens de Production (MRP II) est un système industriel standard pour aider à maximiser l'utilisation des ressources de production. Voir l'annexe D pour une brève introduction et/ou [Arnold, 1998] pour une description détaillée. Le plan de conditions de capacité (CRP) est le sous-plan le plus détaillé de MRP II qui détermine combien de temps il faudra à un centre de fabrication pour produire une quantité et un type déterminés de produits. Ce chapitre explique comment le CRP est employé pour déterminer le temps nécessaire pour fabriquer une quantité et un type indiqués de composant du détecteur ; il démontre comment ces techniques peuvent être intégrées dans CRISTAL, et présente les résultats.

## *8.2 Déterminer combien de temps une production prend en utilisant MRP II*

Le CRP est le sous-plan le plus détaillé de MRP II qui détermine combien de temps il faudra à un centre de fabrication pour produire une quantité et un type indiqués de produit. Le CRP fait ceci en effectuant les trois étapes suivantes:

1. Déterminer les quantités et les types de composants exigés.
2. Déterminer les lignes de production requises.
3. Trouver le chemin critique dans le centre de fabrication.

Les nomenclatures des plans de besoins en matériaux (MRP BoM) sont employées pour déterminer les quantités et les types de composant exigés, et pour relier les lignes de produit ensemble. Ce dernier point est possible parce que la structure de MRP BoM suit la règle qu'aucun type de produit du BoM ne peut être manufacturé jusqu'à ce que tous ses composants soient prêts. Ceci crée des types de montage partiel pour assurer que les lignes de produit sont reliées d'un bout à l'autre. Pour montrer comment ceci fonctionne, considérons l'exemple de fabrication d'une table. La figure 39 montre la conception de la table.

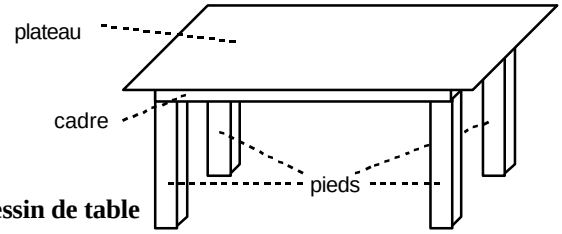



**Figure 39 Exemple de dessin de table**

La table a quatre pieds, un cadre, et un plateau. La figure 40 montre la structure correspondante de composants du produit (PBS).

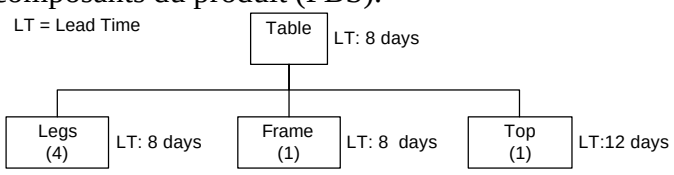



**Figure 40 PBS pour une table**

Imaginons que les itinéraires suivis par les composants de la table dans le centre de fabrication sont comme suit:

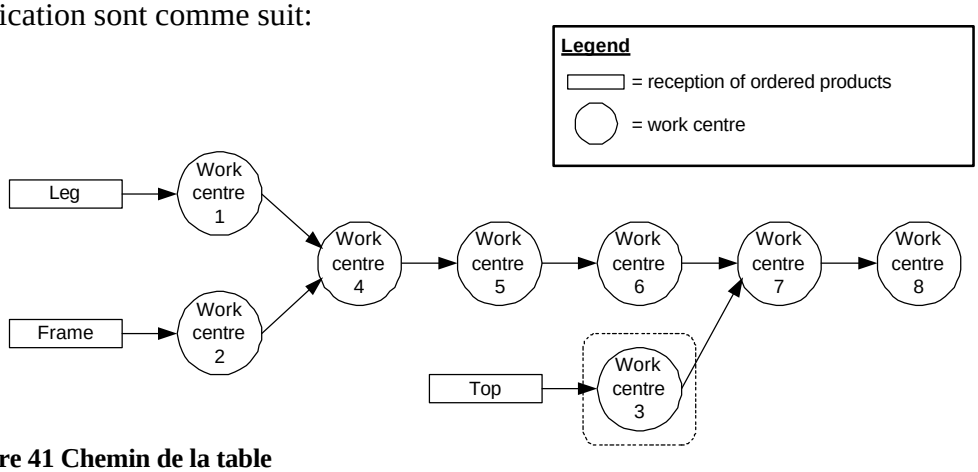



**Figure 41 Chemin de la table**

Les pieds sont visuellement examinés au centre 1, les cadres au centre 2. Tous les deux sont collés ensemble au centre 4. Au centre 5 les pieds et les cadres sont peints, et au centre 6 ils sont vernis. Après avoir été visuellement examiné au centre 3, les dessus de table rejoignent les pieds et les cadres au centre 7 où la table entière est assemblée. Pour finir, les tables réalisées sont visuellement examinées au centre 8. La figure 43 montre la MRP BoM correspondante.

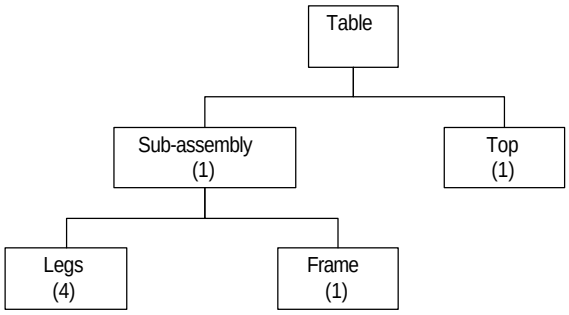



**Figure 42 Table MRP BoM**

En comparaison de la table PBS, la MRP BoM a présenté un type de montage partiel composé de pieds et de cadres. La figure 43 montre ce type de montage partiel superposé aux routes.

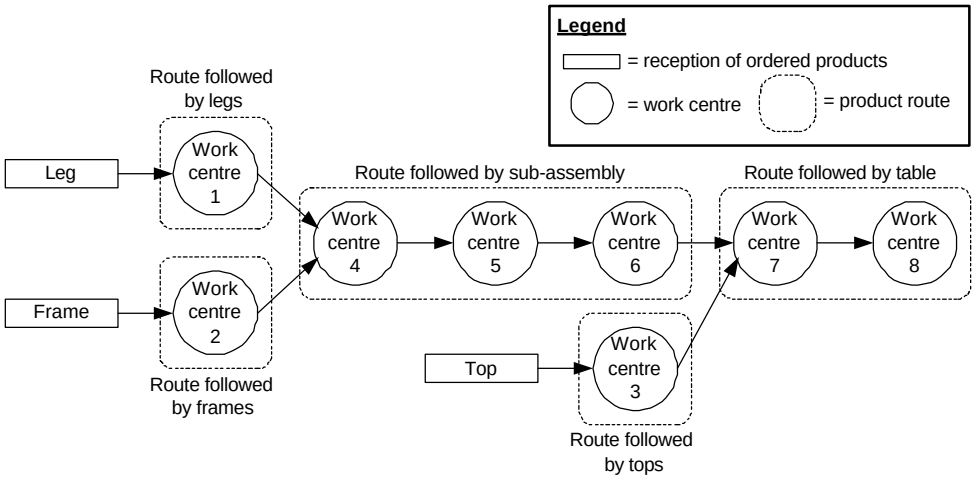



**Figure 43 Travaux de sous-assemblage superposés au chemin suivi par la table**

Une fois que le CRP a déterminé tous les itinéraires de produit, il utilise la capacité et les programmes de travail de chaque poste sur le chemin pour programmer les heures de travail

exigées pour fabriquer la quantité et le type de produit en question. Le temps de fin du dernier poste de travail donne le temps qu'il faudra pour fabriquer la quantité et le type de produit en question.

## *8.3 Intégrer le CRP dans CRISTAL*

Cette section montrera une voie possible pour intégrer les techniques du CRP dans CRISTAL. Elle montrera d'abord comment intégrer les structures de données nécessaires, et puis comment calculer le temps que cela prend pour produire un type et une quantité données de composants du détecteur.

### 8.3.1 Intégrer la notion de centre de travail dans CRISTAL

Pour déterminer combien de temps cela prend de fabriquer une quantité et un type indiqués de produit, la capacité de chaque centre de travail doit être mesurée. Pour le système CRISTAL, la capacité sera mesurée en composants du détecteur par heure. Pour mesurer la capacité d'un centre de travail il est nécessaire d'enregistrer la durée des activités qu'elle exécute. Cette information est déjà rassemblée par le système de CRISTAL sous forme d'historique de processus pour chaque composant du détecteur. Le point important à noter est que les utilisateurs donnent à CRISTAL les informations qu'il faut pour mesurer la durée des différentes activités, c'est à dire que les utilisateurs disent au système quand ils commencent et terminent une activité.

La définition d'un centre de travail pour MRP II est un emplacement sur l'itinéraire d'un produit dans la production, cet emplacement étant un groupe de machines et/ou d'ouvriers qui exécutent le même type d'activité avec la même capacité. La capacité de CRISTAL pour mesurer la durée de chaque activité individuelle lui permet d'assouplir la définition d'un centre de travail, d'être juste un emplacement sur l'itinéraire d'un produit dans la production. Savoir la durée de chaque activité individuelle permet à CRISTAL de mesurer la capacité d'un centre de travail en fonction du type d'activité et du type de produit. L'industrie ne peut pas élargir la définition MRP II d'un centre de travail de cette façon, car en général elle ne suit pas chaque activité individuelle de production. On peut argumenter que l'industrie n'effectue pas ce niveau de la surveillance, car elle n'ajoute aucune valeur au produit final. C'est le contraire pour la construction d'une grande expérience de physique, où l'historique et les caractéristiques de chaque composant du détecteur doivent être enregistrées pour permettre l'étalonnage du détecteur et pour aider dans la tâche de maintenance du détecteur.

En conclusion, CRISTAL n'a pas besoin de restreindre la définition d'un centre de travail à l'exécution d'un seul type d'activité. Un centre de travail peut être capable d'exécuter beaucoup de types d'activité avec différentes capacités. CRISTAL mesurera donc la capacité d'un centre travail en fonction du type d'activité et du type des composants du détecteur.

### 8.3.2 Intégration du fichier principal de centre de travail dans CRISTAL

Le CRP enregistre la capacité du centre de travail dans un "fichier principal de centre de travail". On montrera que CRISTAL peut être étendu à la mesure et à l'enregistrement en temps réel de la capacité démontrée de chaque centre de travail dans une usine.

CRISTAL enregistre la période d'exécution, le nom de l'opérateur et le résultat de chaque activité de production exécutée sur chaque composant du détecteur. Si le nom du centre de travail où l'activité a eu lieu étaient également enregistrés, alors CRISTAL pourrait mesurer la capacité réelle de chaque centre de travail et donc construire un fichier de centre de travail de CRP.

Quand un opérateur ou un gestionnaire local de centre se connecte au système CRISTAL ; ils devraient non seulement fournir leur nom, fonction, et mot de passe, mais également le centre de travail où ils sont postés. Chaque fois qu'une activité est commencée, finie, ignoré ou réinitialisée, l'interface opérateur enverra l'information (« événement ») correspondante du centre de travail au reste du système CRISTAL. L'événement indiquera le nom du centre de travail, le type et l'identification de l'activité, et le type et l'identification (code à barres) du produit en cours de traitement. Le système pourrait alors créer une information semblable à celle du fichier principal de processus. La figure 44 montre une classe possible pour des événements de centre de travail.

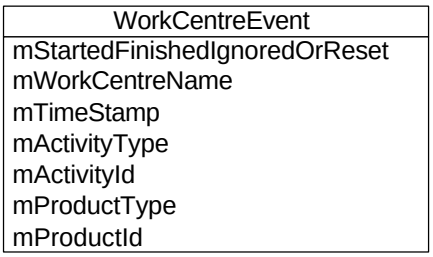



**Figure 44 Evénement de Centre de Travail**

L'attribut mStartedFinishedIgnoredOrReset indique si l'événement est le résultat de l'action commencer, terminer, ignorer ou réinitialiser une activité.

L'endroit idéal pour enregistrer le fichier de centre de travail de CRP est dans l'objet local de gestionnaire de centre (LCM). Le temps pris pour accomplir une activité dépend de trois facteurs: le centre de travail, le type d'activité, et le type du produit. Pour chaque centre de travail on enregistrerait une table bidimensionnelle indiquant les capacités mesurées du centre de travail en fonction du type d'activité et du type de produit comme représenté sur La figure 45.

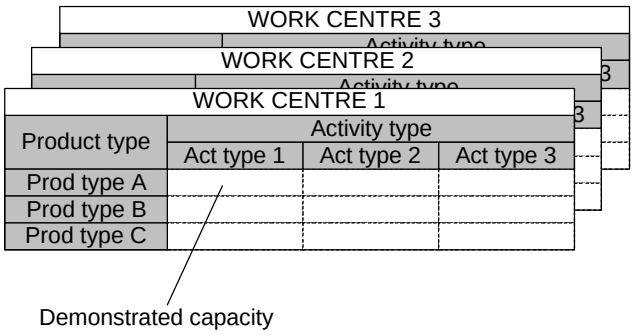



**Figure 45 Tables de capacité d'un centre de travail**

La capacité mesurée sera représentée par deux nombres entiers: le nombre total d'heures passées pour exécuter le type d'activité spécifié sur le type de produit choisi, et le nombre total d'activités exécutées dans ce temps. La moyenne peut être trouvée en divisant le premier par le dernier, et chaque fois qu'on exécute une nouvelle activité, sa durée est ajoutée à toute la période et à nombre d'activités incrémentées d'un.

Le LCM utilisera les événements du centre de travail qu'il reçoit de l'interface opérateur pour garder un enregistrement de l'état actuel de chaque centre de travail. Un centre de travail peut potentiellement travailler à beaucoup d'activités et de produits en même temps. Par conséquent, pour chaque activité de chaque centre de travail, le LCM se rappellera le type de l'activité, le type du produit sur lequel on travaille, et le temps auquel l'activité a été commencée. La figure 46 montre les tables que le LCM utilisera pour se rappeler l'état du centre de travail.

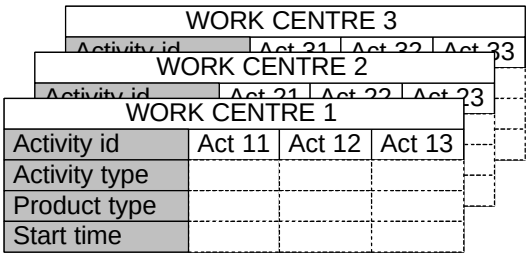



**Figure 46 Tables d'état de centre de travail**

Quand le LCM reçoit un événement « activité démarrée » cela crée une entrée pour l'activité dans la table d'état de centre de travail appropriée. Quand le LCM reçoit l'événement « terminé » correspondant, il calcule la durée de l'activité, emploie le résultat pour mettre à jour la table de capacité de centre de travail appropriée, et puis efface l'entrée d'activité de la table d'état de centre de travail car l'activité n'est plus activée. Si le LCM reçoit une information « activité ignorée » ou « réinitialisation », il retire simplement l'entrée d'activité correspondante de la table d'état de centre de travail appropriée.

Le LCM sera responsable de mesurer des temps de déplacement entre les centres de travail. Le temps de déplacement dépend des centres de travail de départ et de destination et également du type de produit étant déplacé. Par conséquent, le LCM aura une table pour chaque centre de travail, indiquant les temps moyens de déplacement pour chaque type de sous-produit paramétrisé par centre de travail de destination. La figure 47 montre les tables.

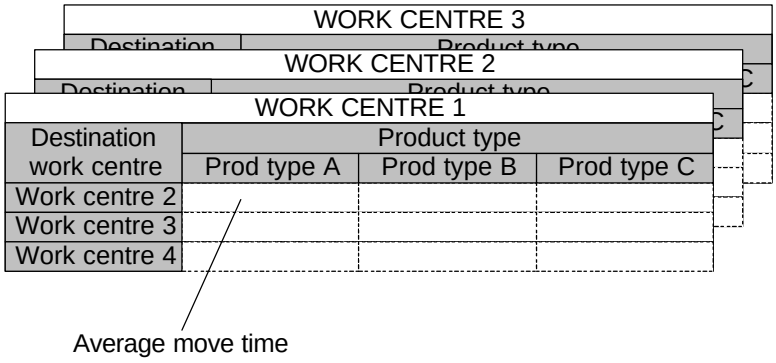



**Figure 47 Tables de déplacement de centres de travail**

Comme la capacité mesurée d'un centre de travail, le temps moyen de déplacement sera représenté par deux nombres entiers: le nombre total d'heures passées à se déplacer et le nombre de déplacements faits.

Pour mettre à jour les tables de déplacement de centre de travail, le LCM doit également mémoriser quels produits sont dans quel centre de travail, et quand on a exécuté la dernière activité sur ces produits dans ces centres de travail. Par conséquent le LCM utilisera les tables d'inventaire suivantes à cette fin:

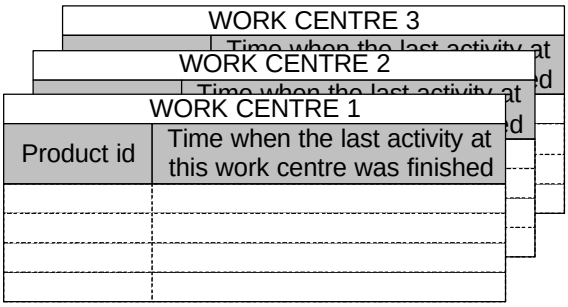



**Figure 48 Tables d'inventaire de centre de travail**

Le LCM utilisera les événements « activité démarrée » envoyés par les interfaces opérateur pour mettre à jour les tables d'inventaire de centre de travail. Quand le LCM reçoit un événement « activité démarrée », il recherchera la table d'inventaire de chaque centre de travail pour trouver dans quel centre de travail se trouve le produit. Il y a trois résultats possibles de cette recherche:

1. Le produit n'est pas trouvé, indiquant que c'est la première activité démarrée sur le produit à ce centre de production.
2. Le produit se trouve au même centre de travail, indiquant que le produit ne s'est pas déplacé. Il est tout à fait normal que plusieurs activités soient exécutées sur un produit dans un centre de travail avant qu'il se déplace à un autre.
3. Le produit se trouve dans un autre centre de travail, indiquant que le produit s'est déplacé.

Si le produit s'est déplacé, alors le LCM calcule son temps de déplacement en soustrayant l'heure de la fin de la dernière activité exécutée au centre de travail de départ de l'heure de démarrage de l'activité actuelle au centre de travail de destination. Le LCM emploie le résultat pour mettre à jour la table de déplacement du centre de travail de départ. La durée du déplacement sera ajoutée au temps total des déplacements, et le nombre de déplacements sera incrémenté d'un. Le LCM retirera également le produit de la table d'inventaire du centre de travail de départ et l'ajoutera à la table d'inventaire du centre de travail de destination.

En conclusion, en ajoutant la capacité du centre de travail, l'état, le déplacement, et les tables d'inventaire au LCM on permet à CRISTAL de déterminer la capacité de chaque centre de travail, et les temps de déplacement des composants du détecteur entre ces centres de travail.

### 8.3.3 Intégration des fichiers d'itinéraire de produit dans CRISTAL

Les fichiers d'itinéraire sont une entrée du CRP. Ils indiquent la séquence complète d'activités à exécuter sur un type de produit, et où ces activités doivent avoir lieu. Chaque type de composant du détecteur de CRISTAL a une définition de processus associée qui indique l'ordre partiel des activités qui doivent être suivies afin de fabriquer des composants de ce type. L'ordre n'inclut aucune information au sujet des centres de travail qui devraient être

utilisés. En fait la même définition de processus peut être utilisée dans différents centres de production, chacun avec son propre ensemble de centres de travail. En conséquence, chaque centre de fabrication doit définir localement son propre ensemble d'itinéraires de produit. Ces itinéraires doivent cependant obéir à la séquence partielle établie par la définition de processus. L'itinéraire pris par un produit dans un centre est "normalement" une séquence unique de centres de travail, c'est à dire qu'il n'y a aucun routage alternatif. Le mot "normalement" a été utilisé ici, car habituellement des routages alternatifs ne sont nécessaires que dans des circonstances exceptionnelles, par exemple quand un centre de travail est en panne. On prétend que la production d'un détecteur est organisée de telle manière que les physiciens puissent facilement décider quels routages alternatifs à prendre quand les choses tournent mal, sans besoin d'ordinateur. La figure 49 montre un exemple de fenêtre pour créer un itinéraire de produit dans un centre de fabrication.

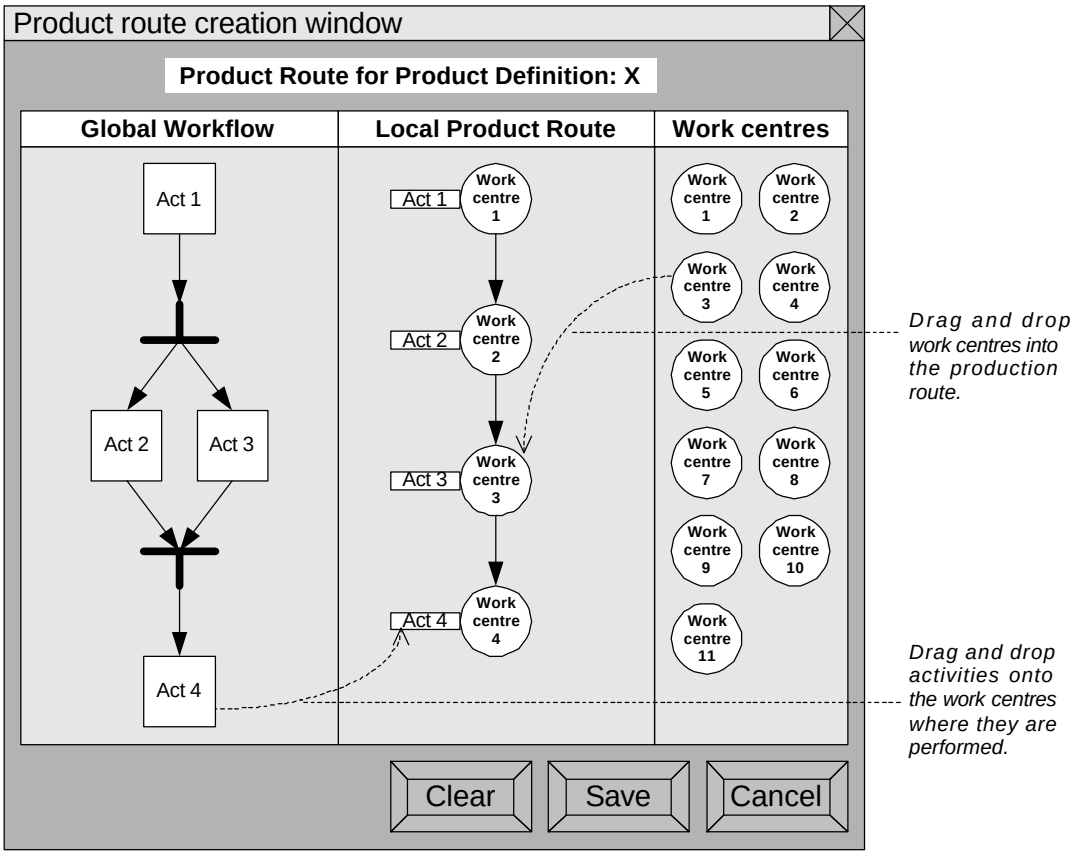



**Figure 49 Fenêtre de création d'itinéraire de produit**

La fenêtre ne devrait pas permettre à des utilisateurs de créer les itinéraires de produit qui n'obéissent pas aux contraintes de la définition de processus.

### 8.3.4 Intégration des agendas de centres de travail dans CRISTAL

Le CRP gère un agenda, c'est à dire une liste des travaux pour chaque centre de travail, pour pouvoir enregistrer quand le centre a du travail travail et quand il est disponible. CRISTAL a besoin également de ces agendas pour la même raison. Le LCM est encore l'endroit idéal pour les enregistrer.

## *8.4 Détermination de la durée de construction avec CRISTAL*

CRISTAL utilisera les mêmes trois étapes que le CRP pour calculer le temps nécessaire pour produire une quantité et un type donné de produit, c'est à dire:

4. Déterminer les quantités et les types de composants exigés.
5. Déterminer les lignes de production requises.
6. Trouver le chemin critique dans le centre de fabrication.

Cependant en ce qui concerne l'étape 2, le PBS de CRISTAL n'a pas les mêmes types de montage partiel que la MRP BoM et ne peut pas déterminer des itinéraires de produit. Le PBS de CRISTAL crée des types de montage partiel de sorte que les caractéristiques soient associées aux composants du détecteur ou aux groupes de composants du détecteur mesuré. Le MRP BoM crée des types de montage partiel pour assurer des itinéraires de produit connectés de bout en bout. Cette section explique comment cette information est déterminée à partir du PBS de CRISTAL et les conditions de début des activités du processus.

### 8.4.1 Déterminer les composants nécessaires

Pour déterminer les quantités et les types de composants nécessaires il faut extraire la composition du composant du détecteur à fabriquer du PBS de CRISTAL. Ceci requiert de savoir quels composants sont des matières premières pour le centre de fabrication et combien de temps cela prend de commander ces composants. Le résultat sera un sous-graphe du PBS. La figure 50 donne un exemple.

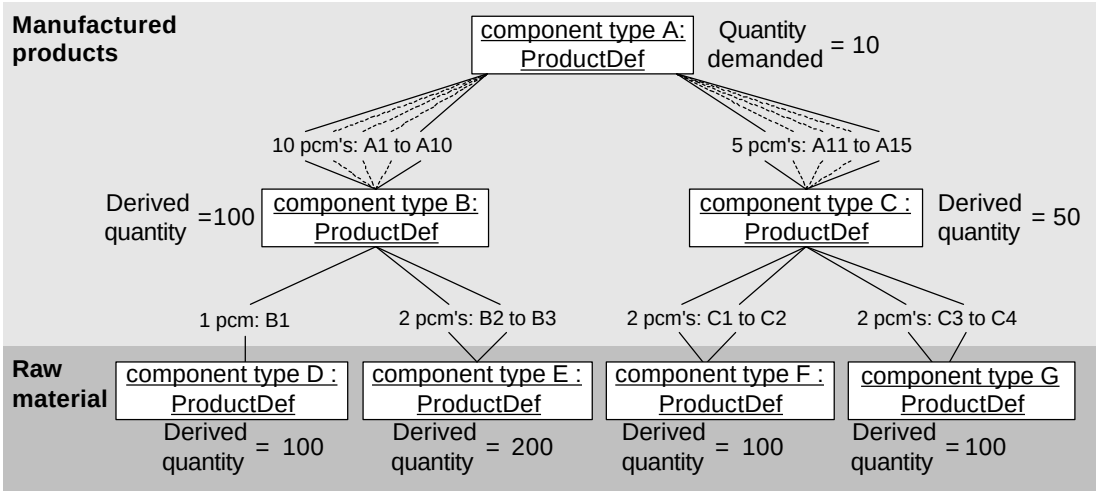



**Figure 50 Composition d'un composant commandé**

Dans cet exemple 10 composants du type "A" ont été commandés. En comptant les objets de composition de produit (PCM) reliant les sous-composants à leurs parents il est possible de déduire combien de sous-composants sont nécessaires. Dans la figure ci-dessus par exemple, 100 composants de type B sont nécessaires car il y a 10 objets PCM reliant ce type au type A.

### 8.4.2 Détermination des itinéraires de construction nécessaires

Comme on l'a dit le CRP emploie la structure de MRP BoM pour relier des itinéraires de produit ensemble, et on ne peut pas faire de même avec la structure du PBS de CRISTAL. Des conditions de début d'activité peuvent être utilisées à la place. Pour chaque itinéraire de

produit impliqué dans la production du composant final du détecteur, on doit effectuer une recherche pour trouver la première fois ou un type de composant fils est exigé comme condition de début d'activité. La figure 51 montre un ensemble d'itinéraires possible pour produire des composants du détecteur du type A.

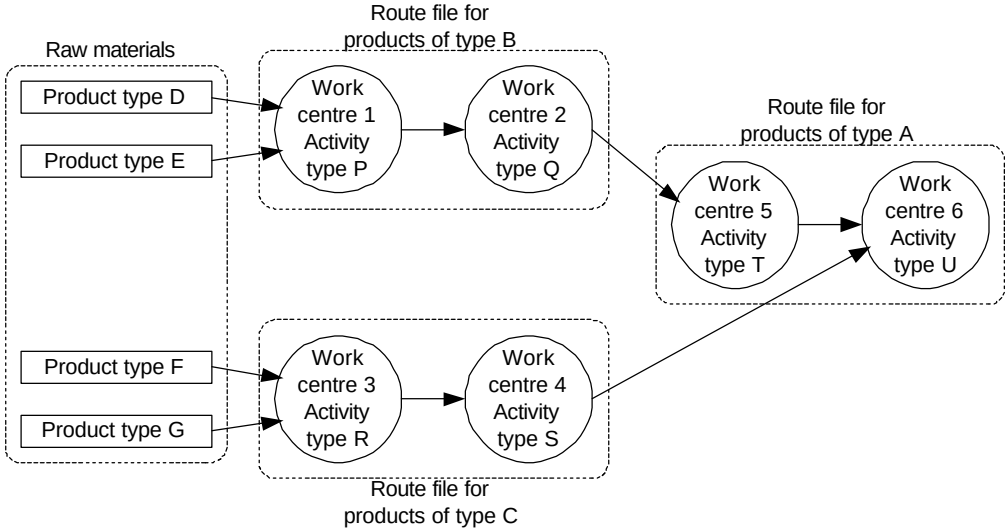



**Figure 51 Itinéraires de produits pour produire des composants de type A**

Les produits du type C rejoignent des produits du type A au centre de travail 6, parce que des produits du type C sont exigés pour la première fois comme conditions de début dans le processus des produits du type A dans les activités du type U.

### 8.4.3 Trouver l'itinéraire critique

Pour trouver l'itinéraire critique, il est nécessaire de faire une programmation anticipée de chaque matière première jusqu'à la première jonction. La programmation anticipée est la technique du fixer la date de début de la production et puis de déterminer la date de finition la plus hâtive. La figure 52 montre les jonctions dans l'exemple d'itinéraire.

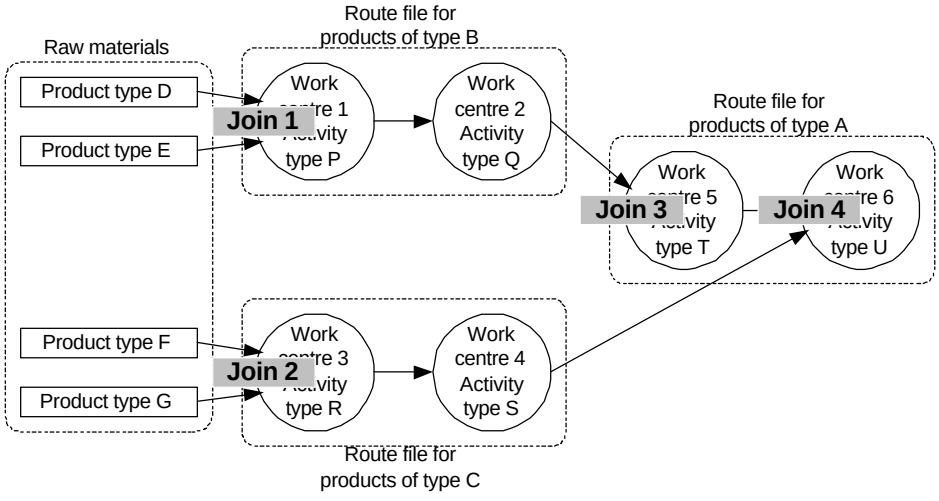



**Figure 52 Les jonctions dans un exemple d'itinéraire**

Les produits du type D et E doivent être programmés à l'avance jusqu'à la jonction 1, et les produits du type F et G jusqu'à la jonction 2. Pour chacun jonction, on marque la date où tous les produits exigés sont prêts. Considérons la jonction 1 par exemple. Si la production doit commencer le lundi 21/08/00, et que cela prend 2 jours pour commander et recevoir des produits du type D et 1 jour pour des produits du type E, alors tous les produits seront prêts

mercredi 23/08/00. En d'autres termes, après l'établissement du programme anticipé, la date à laquelle tous les produits sont prêts à une jonction donnée est la date où les derniers produits arrivent. Une fois les jonctions marquées, on répète l'établissement du programme anticipé jusqu'au prochain ensemble de jonctions, et le processus entier est répété jusqu'à ce que le dernier centre de travail soit atteint. La programmation anticipée des matières premières exige d'ajouter la durée pour commander et recevoir, à la date de début appropriée. La programmation anticipée des produits manufacturés entre les jonctions implique de déterminer le nombre exigé d'heures de travail dans chaque centre de travail sur l'itinéraire, en tenant compte des temps de déplacement, et en créant un agenda pour cet itinéraire. Ceci exige d'extraire les capacités de centre de travail appropriées à partir des tables de capacité de centre de travail, et de diviser le nombre exigé de produits par ces valeurs. La table suivante illustre ce processus pour l'itinéraire entre les jonctions 1 et 3.

| Centre de travail | Type d'activité | Capacité en produits par heure | Nombre de produits requis | Nombre d'heures de travail requises |
|---|---|---|---|---|
| 1 | P | 10 | 100 | 10 |
| 2 | Q | 20 | 100 | 5 |

**Table 4 Heures de travail nécessaires entre les jonctions 1 et 3**

L'agenda de l'itinéraire est construit en consultant les agendas des centres de travail pour déterminer quelles heures sont disponibles pour le travail. La table suivante donne le résultat pour l'itinéraire entre les jonctions 1 et 3.

| | 21/08/00 | 22/08/00 | 23/08/00 | 24/08/00 | 25/08/00 | 28/08/00 |
|---|---|---|---|---|---|---|
| | Lun | Mar | Mer | Jeu | Ven | Lun |
| 8:00 | | | U | W1 | W2 | |
| 9:00 | | | U | W1 | W2 | |
| 10:00 | | | U | W1 | W2 | |
| 11:00 | | | W1 | W1 | | |
| 12:00 | | | W1 | W1 | | |
| 14:00 | | | W1 | U | | |
| 15:00 | | | W1 | W2 | | |
| 16:00 | | | W1 | W2 | | |

Tous produits prêts à la jonction 1 à 8:00 le Mer. 23/08/00

Produits de type B prêts à la jonction 3 à 11:00 le Ven. 25/08/00

**Table 5 Agenda d'itinéraire entre les jonctions 1 et 3**

La lettre "U" indique quand un centre de travail est indisponible. "W1" et "W2" indiquent les heures de travail exigées aux centres 1 et 2 de travail respectivement. L'ordre du jour montre que les produits du type B seront prêts à la jonction3 à 11:00 le vendredi 25/08/00.

La date et l'heure de finition au dernier centre de travail donne le temps qu'il faut au centre de fabrication pour produire la quantité et le type indiqués de composants du détecteur.

## *8.5 Conclusion*

Les fonctionnalités du Plan de Conditions de Capacité (CRP) de MRP II répond à la nécessité pour les physiciens de calculer combien de temps prend la fabrication d'une quantité et d'un type donné de produits. Cette section a montré que ces fonctionnalités peuvent être intégrées dans CRISTAL, en fournissant une solution complète au problème. Il y a deux différences

entre la manière du CRP et de CRISTAL de calculer le temps qu'il faut pour fabriquer une quantité et un type donnés de produit:

1. Le CRP crée des assemblages partiels pour assurer des itinéraires de produit reliés de bout en bout, tandis que CRISTAL joint des itinéraires de produit basés sur les conditions de début des activités exécutées à chaque centre de travail. CRISTAL ne crée pas des types de montage partiel pour assurer des itinéraires de produit reliés de bout en bout, au lieu de cela il les crée pour s'assurer que les caractéristiques sont associées aux composants qui ont été mesurés.
2. CRISTAL peut mesurer la capacité d'un centre de travail en fonction du type d'activité et du type de produit parce que CRISTAL surveille chaque activité individuelle exécutée à un centre de travail. Ceci permet à CRISTAL d'élargir la définition de MRP II d'un centre de travail à un emplacement sur l'itinéraire d'un produit dans la production. Il n'y a aucun besoin de contraindre un centre de travail d'être un groupe de machines et/ou d'ouvriers qui exécutent le même type d'activité avec la même capacité.

En conclusion de ce chapitre, on voit que CRISTAL peut incorporer les fonctionnalités du CRP pour donner aux physiciens la possibilité de calculer combien de temps la production prendra. On peut dire de plus que la solution résultante est plus flexible que celle de MRP II, car la définition des centres de travail a été étendue, et là il n'est aucun besoin de créer des types de montage partiel pour assurer des itinéraires de production reliés de bout en bout.

# 9 Conclusions et travail futur

## *9.1 Introduction*

Ce chapitre commence par récapituler les techniques étudiées et développées pendant cette thèse. Il présente ensuite l'état actuel du travail de CRISTAL, suivi d'une discussion sur les applications industrielles possibles des résultats de la thèse. Des remarques sont faites sur la méthodologie itérative prise dans tout le travail. Des directions possibles pour les travaux futurs sont alors présentées, suivi des conclusions de la thèse.

## *9.2 Techniques étudiées et développées par cette thèse*

Cette thèse a étudié et développé des techniques appropriées à:

- modéliser les compositions et les processus des composants du détecteur,
- visualiser l'assemblage du détecteur et la mesure des composants,
- mesurer la capacité des ressources de production dans un environnement de physique.

Le problème principal s'est posé quand la modélisation de la composition des composants du détecteur et des processus a du supporter le caractère unique de la construction du détecteur. Ceci voulait dire que les compositions de composants et les processus évoluaient pendant le procédé de construction. Une approche orientée description a été adoptée, parce qu'elle a permis aux objets représentant les types de composants de détecteur d'évoluer indépendamment des objets représentant les composants physiques individuels. Ceci a donné un modèle précis du monde réel, où à tout instant deux produits ou plus sur la chaîne de production pouvaient avoir le même type et suivre différentes versions de composition et du processus de construction. Comme ce sera discuté plus tard dans cette conclusion, cette approche orientée description peut être utilisée dans d'autres grands projets d'ingénierie, particulièrement pendant leurs étapes de prototypage.

Cette thèse a présenté une représentation graphique pour montrer la séquence partielle des activités d'assemblage et de mesure exécutées pendant la construction du détecteur. Des techniques ont été développées utilisant cette représentation pour établir l'interface utilisateur. On a montré qu'une combinaison des modèles conceptuels d'observateur et de médiateur [Gamma et al, 1995] a fourni une solution adéquate au problème d'évolution fréquente des interfaces utilisateur.

La Planification des Moyens de Production (MRP II) a été étudiée et certaines de ses techniques pour mesurer la capacité de production ont été intégrées dans CRISTAL. Ceci a eu comme résultat un système qui peut déterminer la capacité de production en temps réel et qui est plus flexible que les systèmes MRP II sur lesquels il a été basé. La dérivation en temps réel de la capacité de production est nécessaire en raison de la nature en évolution constante de la construction d'un appareil unique. Le système résultant de CRISTAL est plus flexible que les systèmes traditionnels de MRP II pour deux raisons. Premièrement les centres de travail n'ont pas la contrainte d'exécuter des activités d'un seul type: ils peuvent exécuter des activités de n'importe quel type, parce que CRISTAL surveille le processus de construction au niveau de chaque composant et de chaque activité, donc permet la mesure des capacités au centre de travail d'être faite en fonction du produit et du type d'activité. Deuxièmement il n'y a aucun besoin de modifier la structure de composition de produit (PBS) pour inclure les montages partiels identifiés où un routage de composant se joint à un autre. Cette information

peut être déterminée à partir des conditions de début d'activité indiquées dans les processus de CRISTAL. Comme l'utilisation de modèles orientés description, les techniques présentées dans cette thèse pour mesurer la capacité de production peuvent être appliquées à d'autres grands projets d'ingénierie, particulièrement pendant leurs étapes de prototypage.

## *9.3 Etat actuel du travail de CRISTAL*

La première version du système de CRISTAL est déjà disponible et en service dans CMS, et a enregistré avec succès les différentes caractéristiques et historiques de production des premiers composants du détecteur ECAL. Le modèle de données de CRISTAL présenté dans cette thèse a été entièrement mis en application en ce qui concerne la modélisation des compositions de composants et les processus. L'interface utilisateur pour les constructeurs du détecteur dans l'atelier a été entièrement mise en application en utilisant le travail de modèle de conception présenté dans cette thèse. Cette thèse a présenté une solution théorique complète au problème de surveillance de la capacité de production en temps réel dans un environnement où le composant physique individuel est suivi. Des travaux futurs seront nécessaires pour mettre en application et analyser les résultats de cette solution.

## *9.4 Transfert de technologie*

C'est la première fois qu'un système d'aide à la construction de détecteurs par ordinateur d'usage universel et dimensionnable a été développé pour la physique des hautes énergies. On a estimé le projet CRISTAL assez important pour faire l'objet d'une évaluation pour un transfert de technologie. On peut espérer que certaines des technologies de CRISTAL seront transférées aux entreprises de classe petites à moyennes (PME/PMI). Les acteurs concernés sont:

- CNRS (Centre National de la Recherche Scientifique)
- Agence Economique Départementale de Haute-Savoie
- CERN
- UWE.

L'approche orientée description adoptée par CRISTAL pour modéliser les compositions de composants et les processus peut être utilisée dans l'industrie partout où il y a un besoin de supporter des processus et des conceptions de produits en évolution. Un candidat idéal pour l'usage de cette technologie serait l'étape prototypage d'un grand projet d'ingénierie.

Les techniques étudiées et développées dans cette thèse pour la visualisation de la construction du détecteur fournissent un exemple concret de la façon de contrôler le processus du point de vue du produit physique, par opposition aux types de produit. Cette approche était nécessaire pour la phase de construction d'un grand détecteur de physique, et peut donc être appropriée pour n'importe quel grand produit dans lequel les caractéristiques et l'histoire de chaque composant individuel sont importantes. Les industries cible potentielles incluent l'espace, la santé, et la fabrication de semi-conducteurs. Les techniques développées pour établir les interfaces utilisateur de CRISTAL peuvent être appliquées à n'importe quel problème de développement d'interface utilisateur où les interfaces doivent évoluer fréquemment et avoir un comportement dynamique complexe.

Les techniques développées pour déterminer la capacité de production en temps réel peuvent être appliquées à l'étape prototypage de n'importe quel grand projet d'ingénierie où le procédé de construction évolue continuellement. A nouveau, les industries potentielles sont la santé, l'aérospatiale, et la fabrication de semi-conducteurs.

## *9.5 Réflexions sur la méthodologie de la thèse*

Ceci est une réflexion sur la méthodologie suivie pendant ce travail de thèse. L'approche itérative adoptée a permis aux résultats du travail de thèse d'être près des besoins des utilisateurs de CRISTAL. C'est une stratégie utile en effectuant n'importe quelle thèse dans le contexte d'un vrai projet de développement de logiciel. Les utilisateurs obtiennent le système qu'ils veulent et l'étudiant en thèse gagne en motivation. Cependant cette stratégie doit être utilisée avec prudence. Les exigences des utilisateurs oscillent, les problèmes d'utilisateur peu importants doivent être délaissés ; la durée de vie de quelques problèmes utilisateur sont une fraction de la durée de vie d'un travail de thèse. L'étudiant en thèse doit pouvoir dire non à certaines des exigences utilisateur afin de fixer les bornes de la thèse et pour travailler sur des problèmes suffisamment complexes pour justifier le travail d'une thèse.

## *9.6 Directions possibles pour le futur*

Ainsi qu'il est dit dans la section 9.3, une solution théorique complète a été présentée au problème de mesurer la capacité de production. Cette solution devra être mise en application et testée dans les travaux futurs. Deux autres directions sont à considérer pour les travaux futurs. Il y a premièrement le besoin de supporter le concept de travail avec des groupes de produits, et il y a deuxièmement le besoin d'aider à optimiser les performances du détecteur en automatisant le positionnement des composants dans le détecteur selon leurs différentes caractéristiques.

### 9.6.1 Travail avec des groupes de composants

Les utilisateurs travaillent dans CRISTAL sur composant à la fois. Cependant il est plus normal que les utilisateurs travaillent en termes de groupes de composants. En ajoutant les capacités de travailler avec des groupes de composants on aide les utilisateurs à construire leur détecteur plus rapidement. En particulier les utilisateurs veulent:

- Travailler avec des lots de composants.
- Envoyer des commandes entre centres de production.
- Suivre des composants individuels dans des groupements définis par l'utilisateur.

Les gens et les machines travaillent naturellement avec des séries de composants. Quelques exemples: l'instrument d'ACCOS mesure les cristaux par séries de 20, et les opérateurs transportent les cristaux dans des boîtes de 5 pour réduire la manipulation. Le concept de lots permettra à CRISTAL d'intégrer plus naturellement les modèles de fonctionnement de ses utilisateurs, et de réduire le nombre de requêtes d'utilisateur, c'est à dire que les utilisateurs peuvent interagir avec CRISTAL en termes de lots plutôt que de composants individuels.

En cas d'une perte de connexion avec le système de commande central, un centre de fabrication doit avoir l'information suffisante pour continuer la production locale. Un centre de fabrication n'a pas les ressources informatiques pour avoir une vue complète du processus

entier de construction (approximativement 1 Terabyte). Par conséquent CRISTAL utilise des commandes entre les centres pour assurer qu'il y a une copie locale de toute l'information liée aux composants du détecteur actuellement dans un centre de fabrication. Toutefois les commandes supportées par CRISTAL sont conçues seulement pour identifier quelles données devraient être présentes localement dans un centre de fabrication, par opposition à aider les utilisateurs à travailler efficacement. La structure de commande choisie était une commande par type de composant, chaque commande contenant les codes à barres des composants étant envoyée. Cette conception a quelques inconvénients en ce qui concerne la convivialité pour l'utilisateur. Il arrive souvent que les commandes physiques envoyées entre les centres se composent de types de composants mélangés. Par exemple un centre pourrait être responsable pour la production des capsules par groupes de 10 prêts pour l'assemblage dans des sous-modules, et un autre centre pourrait être responsable d'assembler ces capsules dans des sous-modules. Pour comprendre, imaginons qu'il y a 8 types de capsule par sous-module. L'envoi d'une commande physique des capsules nécessaires pour assembler 20 sous-modules demanderait 8 commandes de CRISTAL, car il y a 8 types de capsule. Ce que l'utilisateur veut vraiment est une commande simple de CRISTAL qui identifie le fait qu'il y a 20 lots de 10 capsules - 1 lot par sous-module. Avoir une relation 1 à 1 entre la commande physique et les commandes de CRISTAL serait plus intuitif, et connaître les lots permettrait à CRISTAL de s'assurer que les capsules destinées à un sous-module sont restées ensemble et ne se sont pas mélangées avec d'autres. Des utilisateurs sont actuellement forcés de travailler d'une manière artificielle, avec pour résultat du temps perdu et le risque de mélanger les composants du détecteur pendant l'expédition. Comme l'introduction des lots, l'amélioration de la structure des commandes entre centre, permettra à CRISTAL de s'adapter plus naturellement aux méthodes de travail des utilisateurs et lui permettra d'empêcher de mélanger les ordres.

Les composants du détecteur sont groupés pour beaucoup de raisons. Si CRISTAL connaissait ces groupes il pourrait aider des utilisateurs à en trouver les membres. Par exemple, les photodiodes à avalanche (APD) sont produites sur des disques. Si des problèmes sont trouvés avec certains APD après que le détecteur ait été construit, il y a un risque élevé que ceux des mêmes disques aient également des problèmes. Par conséquent les utilisateurs gagneraient un temps considérable si CRISTAL pouvait d'abord identifier les disques dont l'APD défectueux est issu, et puis identifier et localiser le reste des APD de ces disques. Donner à CRISTAL la capacité de définir des groupes de composants, permettra à des utilisateurs de localiser les composants plus rapidement.

### 9.6.2 Automatiser le choix de position des composants dans le détecteur

CRISTAL enregistre les caractéristiques et l'histoire de chaque composant individuel pendant la construction du détecteur. Le PBS actuel de CRISTAL indique quels composants sont composés de quoi. Plus précisément la composition d'un composant est indiquée en termes de localisation et types des éléments fils. Si les spécifications de composition incluaient les conditions à réunir par les caractéristiques de ses composants fils, alors CRISTAL serait en bonne position pour aider les constructeurs à choisir les positions les plus optimales pour les composants individuels dans le détecteur.

## *9.7 Remarque finale*

Cette thèse a étudié, a développé et a appliqué des techniques de gestion de production pour l’aide à la phase de construction d'un grand détecteur de physique. C'est la première fois qu'un système informatique d'usage universel, prévu pour le long terme, ouvert, et adaptable a été développé pour la construction du détecteur. Cette thèse représente une autre étape dans le développement des techniques de gestion de production pour un environnement de physique.

# 10 Glossaire

**Atelier de flux (Flow Shop)**
Une manière d'organiser un ensemble de postes de production. Ici les systèmes de production sont groupés selon leur position dans la séquence de production. Une chaîne de production d'automobiles est un exemple type

**Atelier de tâches (Job Shop)**
Une manière d'organiser un ensemble de postes de production. Ici les systèmes de production sont groupés par fonction, c'est à dire que toutes les machines de coupe sont regroupées à un endroit, et de même pour les autres tâches comme le collage, le polissage, etc …

**Centre de travail (Work Centre)**
Un point où s'effectue une tâche sur le parcours d'une pièce dans un atelier de fabrication.

**Charte des matériaux(Bill of Material or BoM)**
Document précisant quels sont les types de produit ainsi que leur composition.

**Composant du détecteur (Detector Composant)**
Toute partie du détecteur dont les caractéristiques individuelles sont nécessaires à la calibration ultérieure. Ainsi un cristal du détecteur ECAL est considéré comme composant, alors que les écrous, goupilles, et rondelles ne le sont pas.

**Données et Métadonnées (Data and metadata)**
Les données sont des quantités connues ou des objets. Les Métadonnées sont des données décrivant les données. Ainsi, la figure d'une base de données relationnelle est une métadonnée qui décrit la structure de la base de données et les relations entre les données qu'elle contient. De même, le type d'une variable dans un programme écrit en C peut être considéré comme une métadonnée puisque cela décrit un caractère commun à toutes les instances de ce type.

**Gestion des ressources de production (Manufacturing Resource Planning MRPII)**
Le livre "Introduction to Materials Management" [Arnold, 1998] explique que "MRPII s'adresse en premier à la gestion des flux dans pièces entrant, traversant, et sortant d'une usine. Son objectif est optimiser l'utilisation des ressources de l'usine et de fournir à l'utilisateur les informations et les services nécessaires. C'est une planification des tâches à effectuer et leur exécution qui doit se faire avec les ressources existantes, qu'elles soient bonnes ou mauvaises."

**Graphe orienté acyclique (Directed Acyclic Graph or DAG)**
Un graphe dont les nœuds sont reliés par des arcs orientés. Le terme acyclique signifie qu'il n'y a pas de boucles, c'est à dire qu'il n'y a pas de chemin du graphe qui passe plus d'une fois par le même nœud.

**Modèle, Métamodèle, et Architecture en Métamodèle multiniveaux (Model, Metamodel and Multi-layer Metamodel Architecture)**
Le dictionnaire "Oxford English reference dictionary" définit le mot modèle comme une description simplifiée (souvent mathématique) d'un système facilitant les calculs et les prévisions. L'auteur interprète la définition en disant qu'il s'agit d'une description simplifiée

d'un système créée pour un besoin spécifique, et qu'on ne peut donc définir un modèle sans savoir à quoi il va servir.

Un Métamodèle est un modèle de description d'un modèle. Par exemple, si un modèle de bibliothèque est écrit en anglais, alors un modèle (une description) de la langue anglaise peut être appelé métamodèle du modèle de bibliothèque, puisque cela décrit le langage dans lequel le modèle est écrit.

Un métamodèle étant lui-même un modèle particulier, on peut faire un métamodèle de métamodèle, ce qui répété à plusieurs niveau donne une architecture en métamodèles multiniveaux

**Schéma d'analyse (Analysis Pattern)**
Modèle de solution pour un problème d'analyse récurrent. Contient les connaissances qui ont été acquises et qui ont évolué au cours du temps. Facilite la réutilisation des logiciels en transmettant l'expertise acquise.

**Schéma de Conception (Design Pattern)**
Modèle de solution pour un problème de conception récurrent. Le modèle contient les connaissances qui ont été acquises et ont évolué au cours du temps. Le but de ce schéma est de transmettre l'expertise qui a été acquise à d'autres problèmes et donc de permettre la réutilisation du logiciel dans d'autres cas.

# 11 Bibliographie

**[Appleton 2000]** Brad Appleton. Patterns and Software: Essential Concepts and Terminology. Last modified February 2000
http://www.enteract.com/~bradapp/docs/patterns-intro.html

**[Arnold, 1998]** J. R. Tony Arnold. Introduction to Materials Management. Third Edition. Prentice Hall. 1998. ISBN 0-13-698870-9

**[Bachy et Hameri, 1997]** G. Bachy and Ari-Pekka Hameri. What to be implemented at the early stage if a large-scale project. International Journal of Project Management Vol. 15, No. 4, pp. 211-218, 1997.

**[Bazan et al, 1998]** A. Bazan et Al. The Use of Production Management Techniques in the Construction of Large Scale Physics Detectors. Proc of the IEEE Nuclear Science Symposium and Medical Imaging conference, Toronto, Canada. November 1998.

**[Berstein et al]** P. A. Berstein, T Bergstraesser, J. Carlson, S. Pal, P Saunders and D. Shutt. Microsoft Repository Version 2 and the Open Information Model. Microsoft Corporation One Microsoft Way, Redmond, WA 98052-6399 U.S.A.
http://msdn.microsoft.com/repository/technical/infosys/default.asp

**[Blaha et al, 1998]** M. Blaha and W. Premerlani. Object-oriented modeling and design for database applications. Prentice Hall. 1998. ISBN 0-13-123829-9

**[Boehm, 2000]** B. Boehm, edited by W. J. Hansen. Spiral development: experience, principles, and refinements. Spiral development workshop. February 2000.
http://www.sei.cmu.edu/cbs/spiral2000/SR08html/SR08.html

**[Cawsey, 1998]** A. Cawsey. The essence of artificial intelligence. Prentice Hall. 1998. ISBN 0-13-571779-5

**[CIMData, 1995]** CIMData. Product data management: the definition. CIMData, Ann Arbor, MI, USA 1995

**[CMS, 1997]** CMS. The Electromagnetic Calorimeter Project. Technical design report. CERN/LHCC 97-33, CMS TDR 4, 15 December 1997

**[CMS, 1998]** CMS outreach brochure: CMS, Compact Muon Solenoid. CMS Collaboration, CERN. July 1998.
http://cmsinfo.cern.ch/Brochures/IntroToCMS.pdf

**[Delamare et al]** C. Delamare, F. Dittus, R. Evensen, M. Ferran, C. Hauviller, B. Faugeras, N.Hoimyr, M. Mottier, S. Petit, T. Petterson, B. Rousseau, H. van der Welde, G. Faber, J. Nikkola. The CEDAR Project. CERN, Geneva, Switzerland

**[Delamare & Petit, 1998]** Ch. Delamare and S. Petit. CDD CERN drawing directory. User's manual. Version 1.2. CERN Geneva, Switzerland. October 1998.

**[DeMichiel et al, 2000]** L. G. DeMichiel, L. Ü. Yalçinalp and S. Krishnan. Enterprise JavaBeans Specification, Version 2.0, public draft. Sun Microsystems. May 31, 2000.

**[Gamma et al, 1995]** E. Gamma, R. Helm, R. Johnson and J. Vlissides. Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, Inc. 1995. ISBN 0-201-63361-2

**[Hamilton, 1997]** G. Hamilton (Editor). JavaBeans API specification. Version 1.01. Sun Microsystems. July 24, 1997

**[Hollingsworth, 1995]** D. Hollingsworth. The workflow reference model. Issue 1.1 19th January 1995. Document number TC00-1003. Workflow Management Coalition, 2 Crown Walk, Winchester, Hampshire, UK.

**[IPDMUG, 1995]** Assembled by members of the IPDMUG (International PDM Users Group). Integrating/Interfacing PDM with MRP II. 1995
http://www.pdmic.com/IPDMUG/wpipdmug.html

**[Kerhervé et al, 1997]** B. Kerhervé and O. Gerbé. Models for Metadata or Metamodels for Data? Proceedings of the Second IEEE Metadata Conference. September 16-17, 1997.
http://computer.org/conferen/proceed/meta97/papers/bkerherve/bkerherve.html

**[KIF]** Knowledge Interchange Format draft proposed American National Standard (dpANS) NCITS.TS/98-004
http://logic.stanford.edu/kif/dpans.html

**[KQML]** Draft specification of the KQML agent-communication language plus example agent policies and architectures. The DARPA knowledge sharing initiative external interfaces working group.

**[Jennings, 1999]** N. R. Jennings. Agent-based computing: Promise and perils. Thomas Dean (ed.): Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99). Stockholm, Sweden, August 1999. Morgan Kaufmann Publishers.

**[Jennings et al, 2000]** Jennings, N., and Wooldridge, M. "Agent-oriented Software Engineering". In Bradshaw, J. (ed.) Handbook of Agent Technology. AAAI/MIT Press. (to appear) 2000

**[Kerhervé et al, 1997]** B. Kerhervé and O. Gerbé. Models for Metadata or Metamodels for Data?. Proceedings of the Second IEEE Metadata Conference. September 16-17, 1997.
http://computer.org/conferen/proceed/meta97/papers/bkerherve/bkerherve.html

**[Kovacs, 1999]** Z. Kovacs. The integration of product data with workflow management systems through a common data model. Faculty of computer studies and mathematics, University of the West of England, Bristol, England. April 1999.

**[MOF, 1999]** Meta Object Facility (MOF) Specification. Version 1.3 RTF, 27 September 1999. Framingham Corporate Center, Object Management Group, Inc., 492 Old Connecticut Path, Framingham, MA 01701, U.S.A.
http://www.omg.org

**[Muller & Widegren, 1998]** J. Muller & D. Widegren. General user guidelines for the TuoviWDM interface to the EDMS at CERN. CERN, Geneva, Switzerland. 1998.

**[Murray, 1999]** S. Murray. A Software Tool for Optimising the Production of a Large Crystal Calorimeter. AIHENP'99 workshop proceedings: Sixth International Workshop On Software Engineering, Artificial Intelligence, Neural Nets, Genetic Algorithms, Symbolic Algebra, Automatic Calculation April 99 Creta

**[OMG, 1996]** Object Management Group. Product data management enablers. Request for proposal. Version 1.4. OMG Document: mfg/96-08-01. August 1996.

**[OMG, 1999]** The Common Object Request Broker: Architecture and Specification. Revision 2.3.1 October 1999. OMG code: formal/99-10-07
http://www.omg.org/corba/corbaiiop.html

**[Patel et al, 1999]** B. Patel, R. Rusack, P. Vikas, Y.Musienko, S. Nicol, S. Reucroft, J. D. Swain, K.Deiters, D. Renker, T. Sakhelashvili. Avalanche photodiodes for the CMS electromagnetic calorimeter. Presented at the Fifth Workshop on Electronics for LHC experiments. CERN/LHCC/99-33 (1999) 203.

**[Parunak, 1995]** H. Van Dyke Parunak. Autonomous Agent Architectures: A Non-Technical Introduction. Industrial Technology Institute. 1995.
http://www.erim.org/~van/nontech.pdf

**[PDMIC]** PDM Information Center. Understanding product data management
http://www.pdmic.com/undrstnd.html

**[Pikosz & Malmqvist, 1996]** P. Pikosz & J. Malmqvist. Possibilities and limitations when using PDM systems to support the product development process. Proceedings of NordDesign'96, pp 165-175, Espoo, Finland. 1996
http://www.mvd.chalmers.se/publications/pub_list.html

**[Rumbaugh et al, 1991]** J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen. Object-oriented modeling and design. Prentice Hall. 1991. ISBN 0-13-630054-5

**[Smith, 1980]** R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE transactions on computers. Vol. C-29, No. 12, December 1980

**[Sommerville, 1992]** Ian Sommerville. Software engineering. 4th edition. Ian Sommerville Addison and Wesley publishing company. 1992. ISBN: 0-201-56529-3

**[Sorensen, 1995]** R. Sorensen. A comparison of software development methodologies. Software Technology Support Center. January 1995.
http://www.stsc.hill.af.mil/CrossTalk/1995/jan/Comparis.asp

**[Thomas, 1998]** A. Thomas. Enterprise JavaBeans technology, server composant model for the Java platform. Prepared for Sun Microsystems, Inc. by the Patricia Seynold Group. Revised 1998.

**[UML, 1997]** UML Semantics. Version 1.1, 1 September 1997. Rational Headquarters, Rational Software Corporation, 18880 Homestead Rd, Cupertino, CA 95014
http://www.rational.com/uml

**[Weiss, 1999]** Gerhard Weiss (editor) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press. 1999. ISBN 0-262-23203-0

**[Williams, 1994]** C. S. Williams. What is product data management and why should I care? The Sun Observer, pp. 25-27. November 1994.

**[Wooldridge, 1997]** M. Wooldridge. Agent-based software engineering. IEE Proc Software Engineering 144:26-37. 1997.

# 12 Appendix A – Introduction to CMS

This appendix is intended for readers with no prior knowledge of high energy physics accelerators and detectors. It explains what is the Compact Muon Solenoid (CMS) detector, starting from a simple television set.

A television is a very simple accelerator in the form of a glass vacuum tube, as shown in figure 53. The screen is at one end of the tube, and a hot wire, like the filament of a light bulb, is at the other. This wire acts as a particle source, boiling off negatively charged electrons into the vacuum. These are then accelerated towards a positively charged electrode. The electrode is a hollow cylinder and some of the electrons pass straight through it and go on to hit the screen. There are some charged plates between the electrode and the screen. These create electromagnetic fields that bend the beam of accelerated electrons, so that it sweeps across the screen to form the horizontal lines of the image. The inside of the screen is covered in phosphor that glows at the points where the electrons strike it. A purpose built particle accelerator is similar to a television. It has a particle source, electromagnetic fields to deflect particles, and a particle detector (the screen in the case of the television). However a purpose built particle detector has many accelerating electrodes.

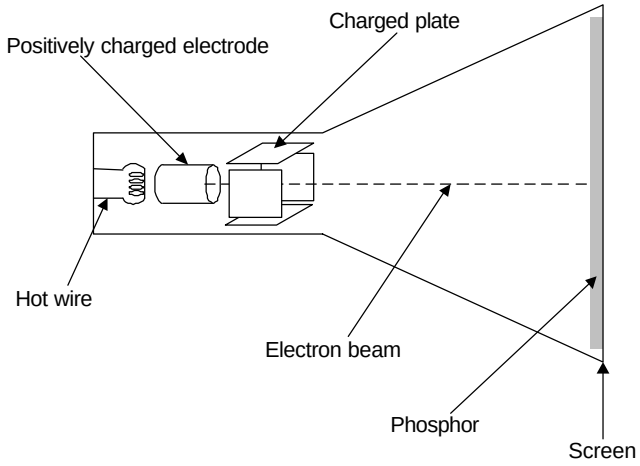



**Figure 53 A television is a simple accelerator**

There are two types of particle accelerators, linear and circular. A LINear Accelerator (LINAC) accelerates charged particles in a straight line. Figure 54 shows a LINAC with an energy of 50 MeV. To accelerate particles to the energies needed by physicists requires extremely long accelerators. Circular accelerators overcome this problem by allowing the beam to be accelerated faster and faster in a circle. Two other advantages of circular accelerators are:

1. They can be used to store particle beams, for example to compensate for a particle source with low productivity.
2. They can accelerate two beams in opposite directions, which can then be directed into each other, thus making the accelerator act as a collider.

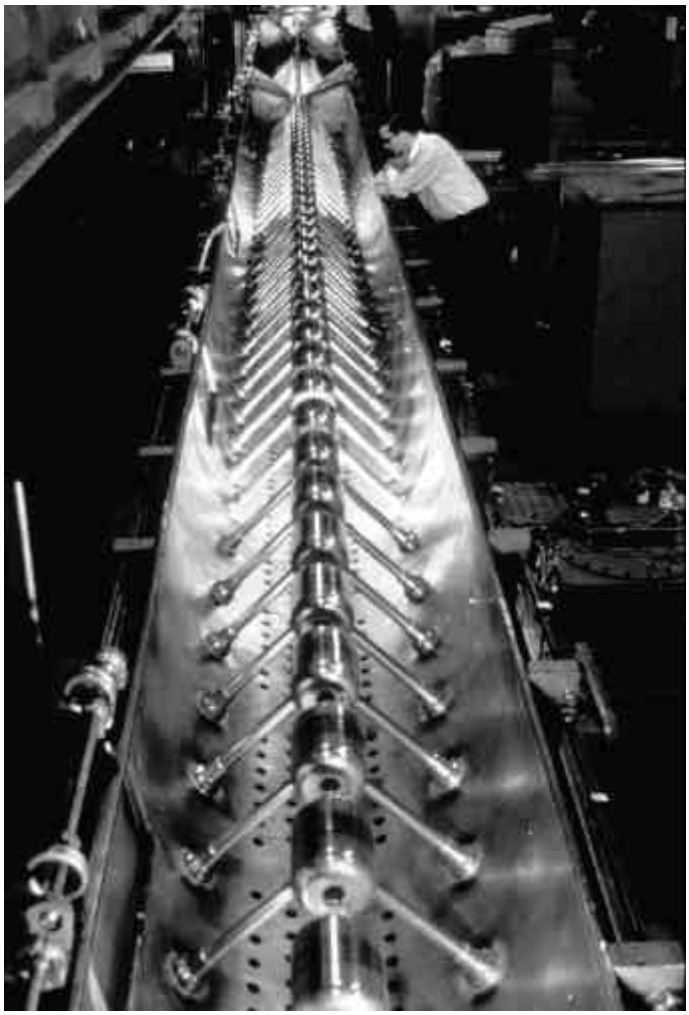


**Figure 54 Linear proton accelerator.**

CERN photo. © CERN Geneva.

CERN (Conseil Européen pour la Recherche Nucléaire) is a world leading particle physics laboratory, representing Europe's interests in pure physics. Before the creation of CERN the field of physics was lead by the Americans. No single European country could compete by itself, so the particle physics laboratory of CERN was founded in 1954 and was one of Europe's first joint ventures. 20 European member states are presently involved. CERN looks at the composition of matter and at the forces that hold it together. It provides some 6500 scientists, half of the world's particle physicists, with state-of-the-art scientific facilities. CERN's scientists represent 500 universities and over 80 nationalities. CERN has just fewer than 3000 staff who provide state-of-the-art scientific facilities for researchers to use. These include particle accelerators and detectors. The accelerators accelerate tiny charged particles including protons, electrons and ions, to speeds close to that of light, and then let them collide with other particles. The detectors examine the results of these collisions. Physicists are studying particles, as they are the building blocks of matter.

There are 10 accelerators at CERN, forming the most versatile accelerator complex in the world. Figure 55 shows how CERN's accelerators are interconnected.

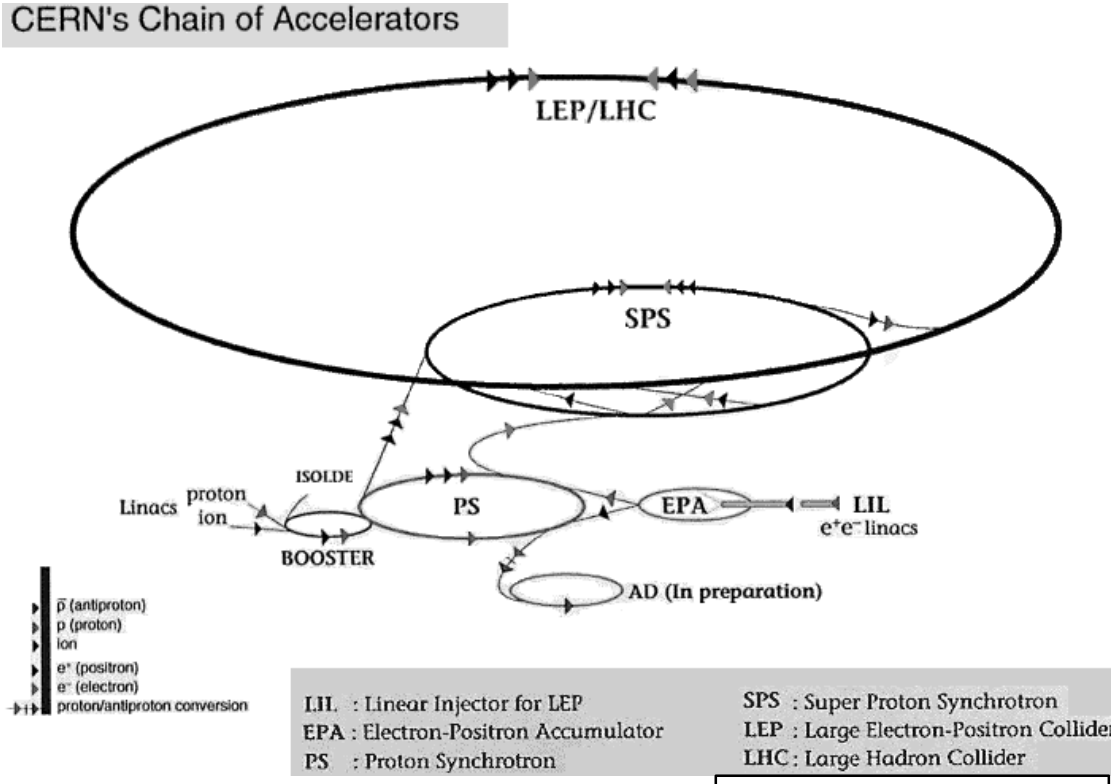


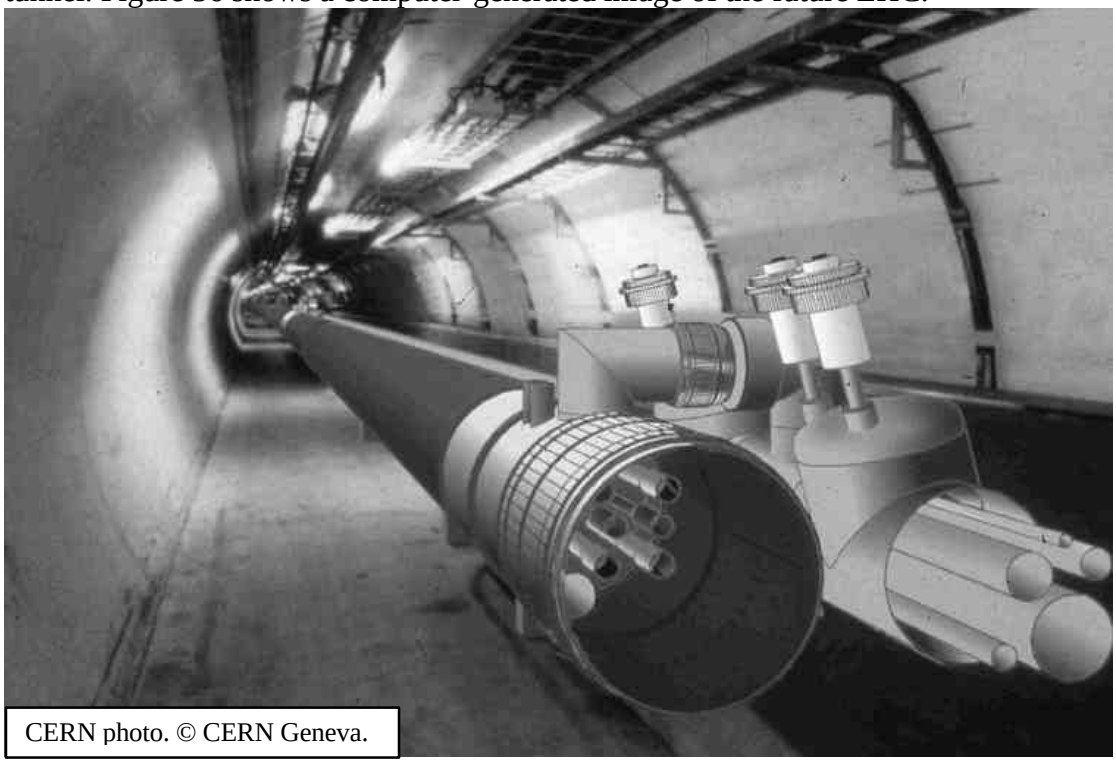

**Figure 55 CERN's chain of accelerators**



The largest accelerator at CERN, which is also the largest in the world, is the Large Electron Positron collider (LEP). Buried approximately 100 metres underground, LEP is a circular accelerator with a circumference of 27km. As its name suggests it collides electrons with positrons (anti-electrons). LEP has four symmetric collision points around its ring, where large detectors observe the results. When an electron and positron collide, they destroy each other and produce energy. This energy changes almost immediately back into particles, just like matter must have formed from energy in the early stages of the Universe. The "Standard Model" models particles and their interactions. It is our best description of sub-atomic Nature. It predicted the existence of the $Z^0$, $W^+$ and $W^-$ particles. Physicists have been using LEP to and create and detect these particles to test the Standard Model. LEP started operation in the summer of 1989. For the 6 years that followed, LEP was tuned exactly to the energy value needed to produce the $Z^0$ particle, the neutral carrier of the weak force. The energy was almost doubled in the autumn of 1995, and in the summer of 1996 LEP ran at the exact value needed to produce pairs of $W^+$ and $W^-$ particles, the charged carriers of the weak force. Millions of $Z^0$ particles and hundreds of W particles have been detected, and this has allowed extremely precise tests to be carried out on the Standard Model.

The data produced by LEP have given a preview of discoveries yet to come, and the evidence indicates that new physics, and answers to some of the most profound questions of our time, lie at energies around 1 TeV. To produce such energy levels, a

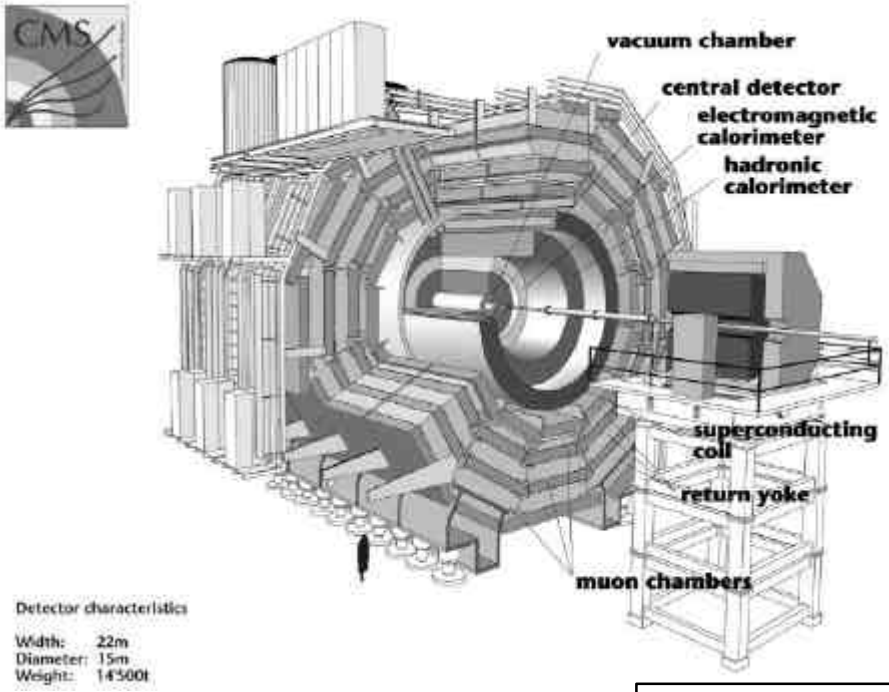
new accelerator called the Large Hadron Collider (LHC) will be built in the LEP tunnel. Figure 56 shows a computer-generated image of the future LHC.



CERN photo. © CERN Geneva.

**Figure 56 Computer generated image of the future LHC**

Together with the new LHC accelerator will be a new set of detectors, one of which will be the Compact Muon Solenoid detector (CMS). Figure 57 gives an exploded view.




CERN photo. © CERN Geneva.

**Figure 57 Compact Muon Solenoid (CMS) detector layout**

CMS is a general-purpose proton-proton detector designed to run at the highest luminosity at the LHC. It will also be able to make useful studies at the initially low

luminosities of the earlier stages of LHC operation. The detector will be optimised for the search of the predicted Standard Model Higgs boson particle over a mass range of 90 GeV to 1 TeV. The main design goals are to construct 4 high performance sub-detectors of complementary functions. Figure 57 shows their position within CMS. Starting at the centre and moving out, there is the central tracker, Electromagnetic CALorimeter (ECAL), Hadron CALorimeter (HCAL), and the muon detection system. CMS is built around a super conducting solenoid magnetic that will produce a 4 Tesla magnetic field inside its coil. The magnet is 13m in length and 3m in radius. The central tracking system, ECAL and HCAL will be housed inside of the magnet so their performance is not affected by coil material. The main objective of the central tracking system is to reconstruct the trajectory of charge particles. The tracker represents a considerable quantity of material in front of ECAL, which makes it harder for ECAL to detect photons and electrons. This is unavoidable, as the tracker requires a large number of small detection cells in order to handle the high particle rate of the LHC. ECAL will absorb the energy of photons and electrons and produce signals proportional to the energy deposited. ECAL was chosen to be a homogeneous crystal calorimeter in order to have excellent energy resolution. ECAL will be used to measure the energy and direction of hadron jets. Fine segmentation will be required to separate nearby jets and to measure the jet direction with sufficient precision. ECAL will need to be hermetic to detect and measure an imbalance in traverse energy. The muon system will identify muons and measure their momentum. It will also act as a trigger as new physics is accompanied by the presence of muons of high transverse momentum in the final state. The muon detector is outside of the magnet, as muons do not interact very much with matter.

CMS will have approximately 16 million individual detector channels. Powerful computers will synchronise these channels with LHC collisions. LHC will provide CMS with approximately 800 million collisions per second. Head-on collisions will be quite rare and the production of new particles even rarer. One Higgs boson is expected to be produced every $10^{13}$ collisions. With 800 million collisions a second this means one Higgs boson per day. The trigger and acquisition system of CMS must reduce the event rate from $10^9$ Hz down to less that 100 Hz, the maximum rate that can be archived for offline analysis. A staged trigger system selects potentially interesting events to achieve the $10^7$ rate reduction factor. This trigger system is one of the key problems faced by CMS, as it affects all subsequent analysis. Figure 58, copied from [CMS, 1998], shows the CMS trigger and data acquisition system.

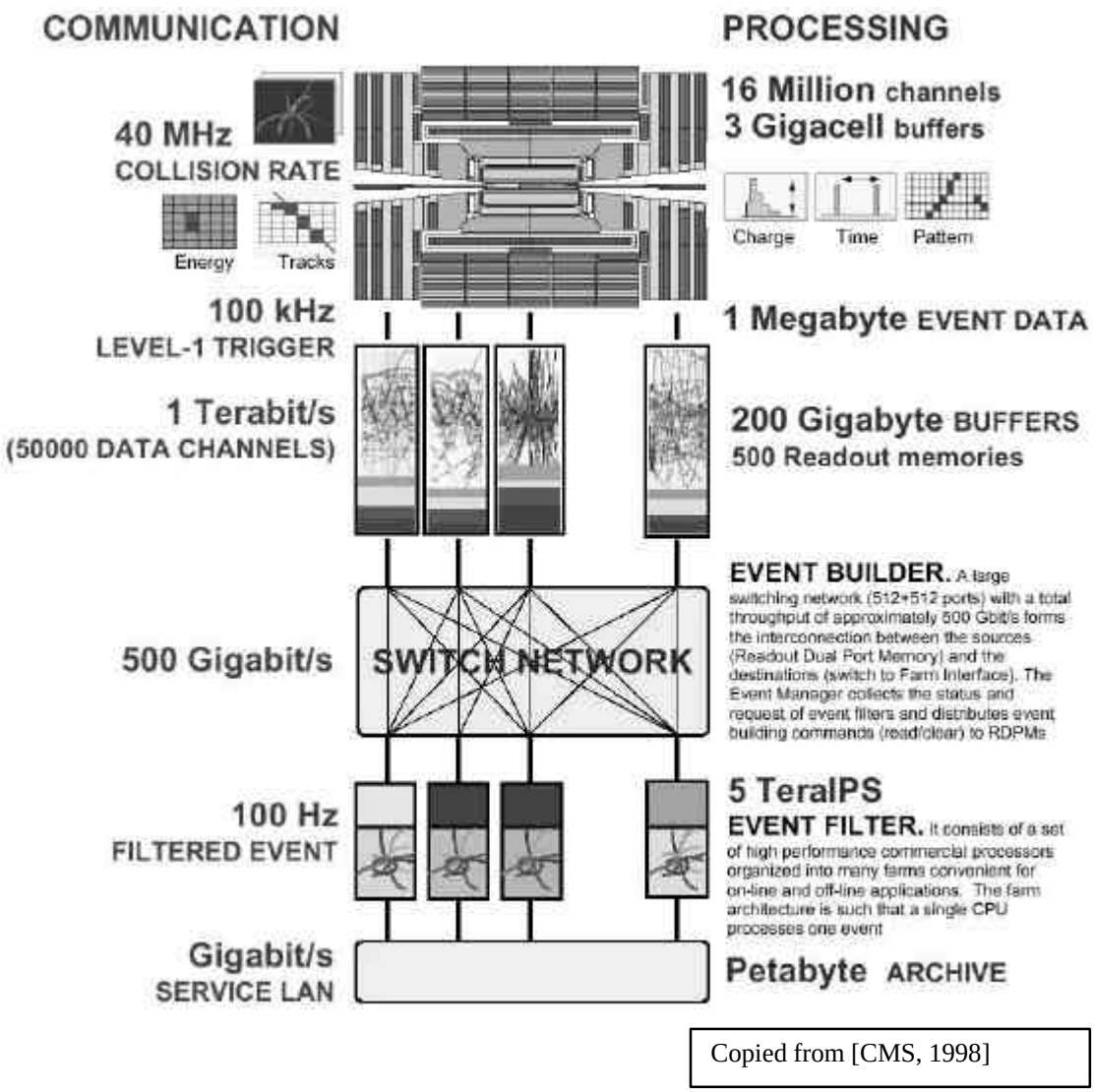



**Figure 58 CMS trigger and data acquisition**

# 13 Appendix B – Importance of ECAL energy resolution

The LHC's search for the Higgs particle will rely strongly on the information produced by ECAL. One of the main objectives of the CMS group was to build an electromagnetic calorimeter, as this type of detector offers the best performance for energy resolution. To help explain the importance of this, it is necessary to understand how ECAL will detect the existence of the Higgs particle.

A Higgs particle will decay immediately after it is produced. In most cases the decay will result in the emission of 2 gamma rays. ECAL will try to detect these rays in order to prove the existence of the Higgs. This is where the need for high performance energy resolution becomes paramount. Besides the gamma rays produced by the decaying Higgs particles, there are also large amounts produced by background noise. Figure 59 helps emphasise that the expected band of energy in which the Higgs gamma rays are to be detected is extremely thin, and that the background production of gamma rays is relatively very high.

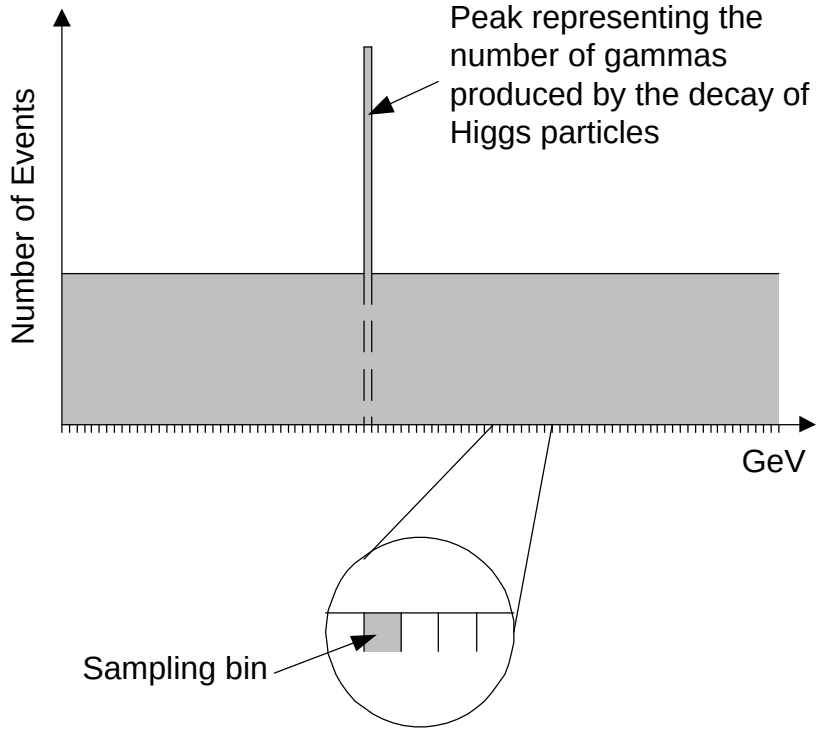



**Figure 59 Event count of Higgs gamma rays with high energy resolution**

If the sample bins are too large, gamma rays from background reactions will be counted with those produced by the decay of Higgs particles. This will result in the hiding of the peak. Figure 60 helps show the effect.

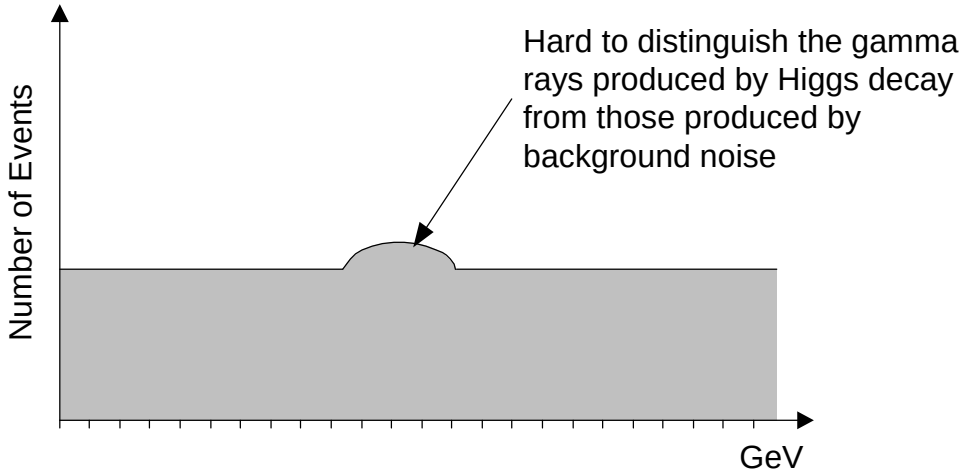



**Figure 60 Hiding the Higgs decay peak**

The hiding of this peak would be disastrous to the success of ECAL. Therefore quality and performance are paramount objectives in the production of ECAL.

# 14 Appendix C – Unified Modelling Language (UML)

## *14.1 Introduction*

The software development methodology used during the analysis and design stages of the CRISTAL project was based on the Object Modelling Technique (OMT), see [Rumbaugh et al, 1991] for more details. At the time analysis and design was being carried out, the Unified Modelling Language (UML) notation became very popular and was adopted by the CRISTAL team. There are hardly any differences between the UML notation and the older OMT notation. This is of no surprise, as one of the objectives of UML was to unify the notation of OMT, Booch, OOSE/Jacobson, and other methodologies. To help readers unfamiliar with object-oriented modelling and UML, this section gives a short introduction to the concepts and notation used in this thesis.

OMT is analysing, designing and implementing computer software using objects. An object encapsulates a set of data, called attributes, and the methods that manipulate those attributes. A class defines a set of objects with the same names and types of attributes, and the same methods. An object is said to be an instance of the class that defines it. This is analogous to data types and variables in function based computer languages, such as C and Pascal. For example, the integer variable x containing the value 7 can be said to be an instance of the data type integer.

UML is a graphical notation for modelling complex systems. Models produced by following the OMT methodology can be drawn using the UML notation. As mentioned above, there are not many differences between the UML notation and the original OMT notation. UML supports many diagramming techniques, for some examples: class diagrams, object diagrams, sequence diagrams, and state diagrams. This thesis contains class, object and sequence diagrams using the UML notation. State diagrams are also shown, but these are based on the older OMT notation and are explained in the chapter where they appear. This appendix will introduce the class, object and sequence diagramming techniques used in this thesis.

## *14.2 Object and class diagrams*

A UML class diagram shows classes, objects, and the relationships between them. An object diagram is the special case of a class diagram, where only objects and the relationships between them are shown. To give an example of a class diagram, consider three bank accounts. These could be represented by three objects whose attributes are the amount of money in the account and the present rate of interest, and whose methods are makeDeposit(), makeWithdraw(), and addInterest(). All three objects inherit from the same class, therefore all three have the same attribute names and types, and the same methods. Figure 61 is a UML class diagram showing the three bank account objects and their class.

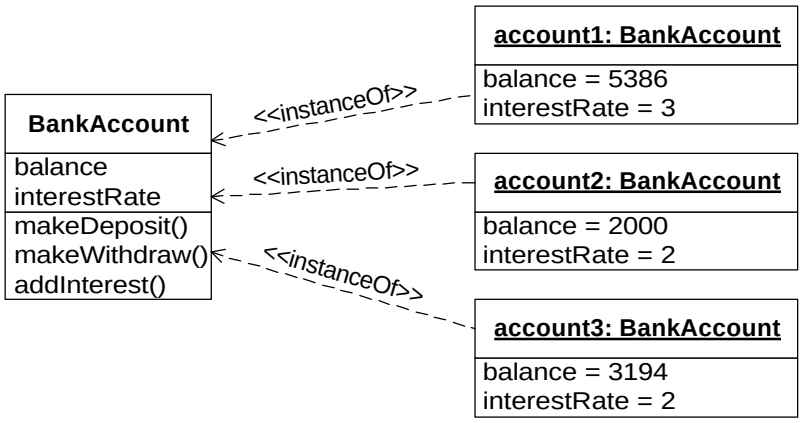



**Figure 61 Example class diagram**

Each class and object is represented by a rectangle. The rectangle of a class can be subdivided into three sections: top, middle, and bottom. The top section contains the name of the class, the middle section the attributes of the class, and the bottom section the methods of the class. The rectangle of an object can be subdivided into two sections: top and bottom. The top section gives the identifier of the object followed by a colon and the name of the class that defines it. The identifier, colon and class name are all underlined. The bottom section gives the attributes of the object and their values. The <<instanceOf>> arrows in the above class diagram show that all three objects are instances of the *BankAccount* class.

### 14.2.1 Links and associations

Objects can be linked together. For example an object representing a person can be linked to a bank account object, and an object representing a bank can be linked to many bank account objects. The following object diagram gives an example, where two people have bank accounts at the same bank, and a third person has no bank account at all.

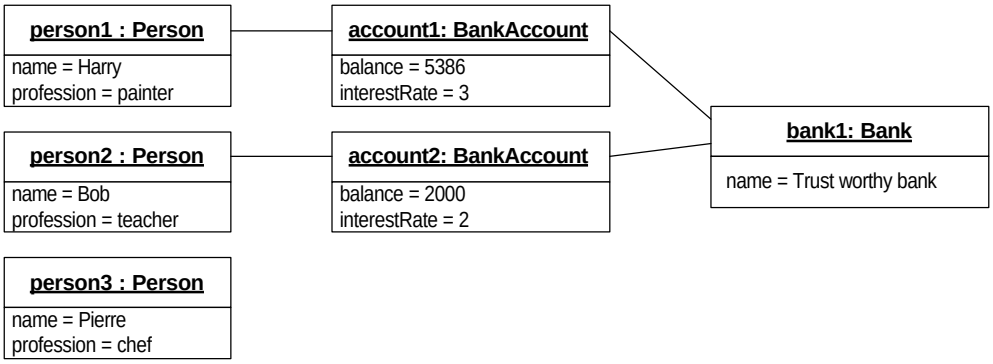



**Figure 62 Links between objects in UML notation**

Links between objects are summarised by associations between their corresponding classes. In technical terms, a link is an instance of an association.

Figure 65 is a class diagram modelling the previous object diagram using classes and associations.

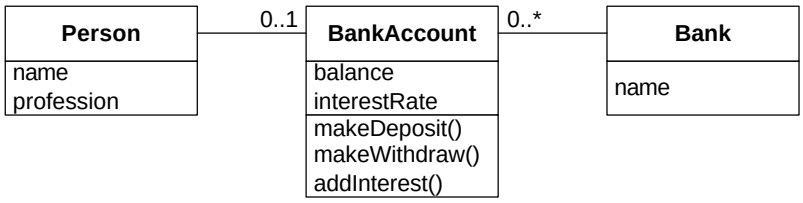



**Figure 63 Modelling objects and links with classes and associations**

The association between the *Person* and *BankAccount* classes models the links between the *person objects* and the *bank account objects.* Likewise the association between the *Bank* and *BankAccount* classes models the links between the *bank object* and the *bank account objects.* The text at the ends of an association indicate cardinality. The following table lists the meanings of the text used to indicate cardinality. Note that an association end with no text means a cardinality of one.

| Text at end of association | Meaning |
|---|---|
| | One |
| 0..1 | Zero or one |
| 0..* | Zero or more |
| 1..* | One or more |
| n..* | n or more |
| n..m | Between n and m |

**Table 6 Cardinality texts and meanings**

Looking at the table and the above class diagram, it can be seen that reading from left to right:

A person has **zero or more** bank accounts.
A bank account must be managed by **one** bank.

And reading from right to left:

A bank can manage **zero or more** bank accounts.
A bank account must belong to **one** person.

Besides normal associations, this thesis also uses the inheritance and aggregation associations.

### 14.2.2 Inheritance

If class B inherits from class A, then class A is said to be the superclass of class B, and class B is said to be a subclass of class A. A superclass can have many subclasses, and a subclass can inherit from many superclasses. A subclass inherits some or all of the attributes and methods of its superclass. An instance of a subclass can be said to be a “type of” of an instance of the corresponding superclass, i.e. an instance of a subclass can be used wherever an instance of its superclass can be used. An

association with a triangle in it shows inheritance. The triangle points towards the superclass. Figure 64 gives an example of the UML notation for inheritance.

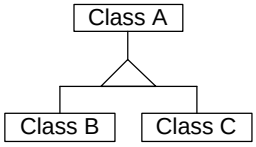



**Figure 64 Example of inheritance**

The above figure says that classes B and C inherit from class A.

### 14.2.3 Aggregation

Aggregation shows which objects are composed of which. On a class diagram, an association with a hollow diamond at one end shows aggregation. Instances of the class at the diamond end of are composed of instances of the class at the straight end. Consider the following class diagram.

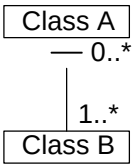



**Figure 65 Example of aggregation**

The diagram says that an instance of class A is composed of one or more instances of class B, and that an instance of class B is a component of zero or more instances of class A. The diagram is saying that instances of class B can be shared by instances of class A. There is a stronger form of aggregation known as composition. The notation is the same except that the diamond is filled. With composition a component cannot be shared between composites.

## *14.3 Sequence diagrams*

A sequence diagram shows a sequence of messages between objects. Figure 66 gives an example.

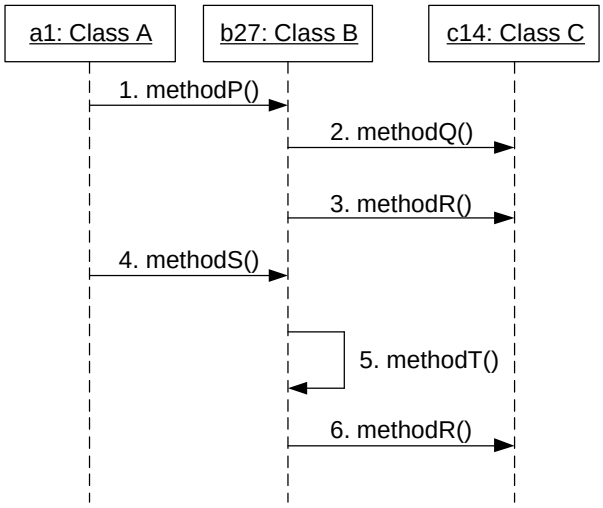



**Figure 66 Example sequence diagram**

All the objects go across the top of diagram, with a vertical time line connected to each. When an object calls a method on another, a horizontal arrow is drawn between their time lines with the name of the method on the arrow. The arrows can optionally be numbered to help see the sequence, especially if there is concurrency. An object can send a message to itself. This is represented by an arrow whose source and destination are the same object time line. This is shown in figure 66 by message number “5.”

# 15 Appendix D - Manufacturing Resource Planning (MRP II)

Manufacturing Resource Planning (MRP II) is a fully integrated manufacturing planning and control system. The "II" part of the abbreviation distinguishes it from Material Requirements Planning (MRP), a sub-plan of MRP II. MRP II strives to balance demand with capacity, to optimise the use of manufacturing resources. MRP II integrates many plans and schedules. This chapter describes some of these to give a general idea of what MRP II is. For a detailed introduction see [Arnold, 1998]. Figure 67 shows the plans and schedules, and the relationships between them.

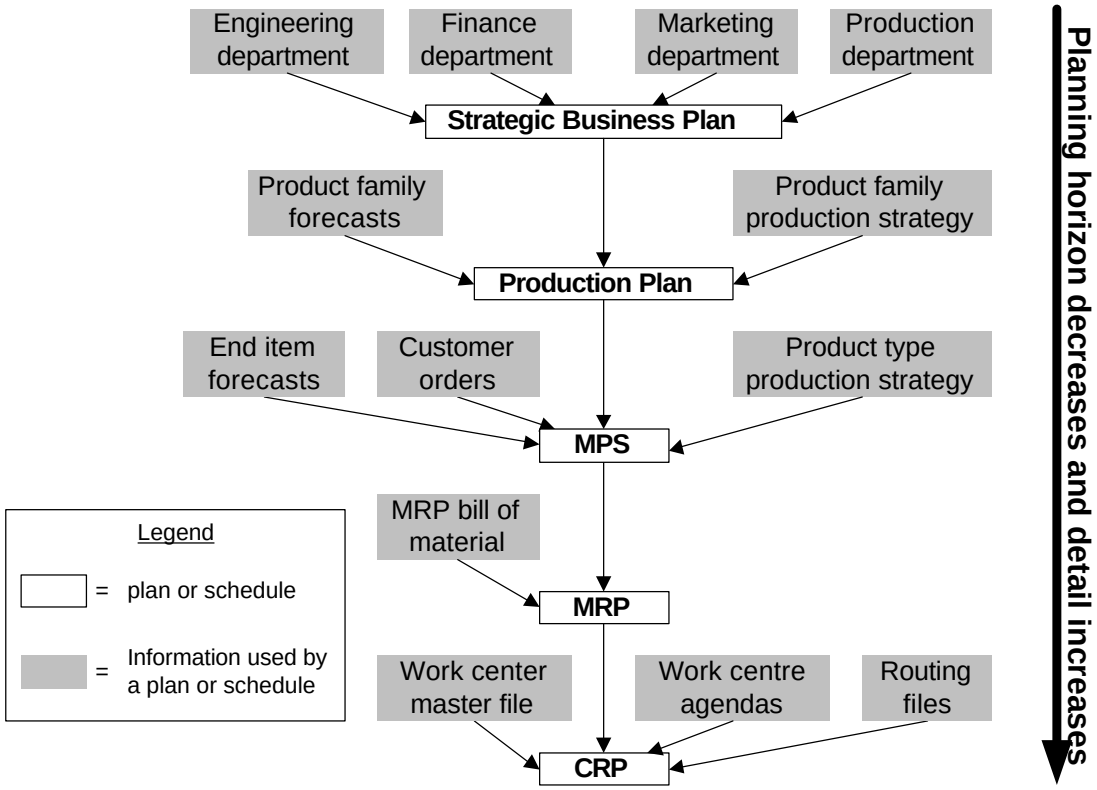



**Figure 67 MRP II sub-plans**

The *strategic business plan* at the top of the figure is long term and low in detail. As you move down the figure, planning horizons reduce and detail increases. The strategic business plan takes input from the engineering, finance, marketing and production departments to produce the overall company goals which are an input to the production plan.

The *production plan* gives the production schedule for the year in terms of product families. (Individual products are grouped by type, and product types are grouped by family. For example a detector component could exist with individual bar code "12345", type "6L" and family "crystal"). The production plan uses product family forecasts, the company goals, and the production strategy for manufacturing product families. The strategy could be based on chasing customer demand, levelling production, or subcontracting. Chasing customer demand means producing products at the exact moment they are needed. This minimises inventory, but causes production to vary. Levelling production means producing products to meet the average demand

each period. Production is constant, but inventory varies. Subcontracting means producing products to meet the minimum demand and subcontracting out all extra work. Production is constant but subcontracting may cost more than internal production.

The *Master Production Schedule (MPS)* says what manufacturing will produce during the year in terms of end items, i.e. the types of product bought by the customer. The MPS works within the limits of the production schedule. For example, if the production schedule says that 1 thousand products of family crystal will be produced during a time period, then the MPS can say that 5 thousand crystals of type A and 5 thousand of type B will be produced during that period. The MPS takes customers orders as one of its inputs and combines them with the demand that was forecast, the planned production of the production plan, and the production strategy for manufacturing product types. There is not a one to one mapping between customer orders and the manufacturing orders produce by the MPS. The MPS knows what will be produced and what manufacturing is capable of. Therefore the MPS can say what capacity is available for the sales department to promise.

Driven by the MPS, the *Material Requirements Plan (MRP)* uses a MRP Bill of Material (BoM) to decide what components need to be manufactured by the factory and when. The MRP BoM is a collection of tree structures that specify how products are put together, in terms of what sub-components are needed and when. End items are found at the roots of the trees. Consider figure 68 - lead time is the average rate at which an item is manufactured.

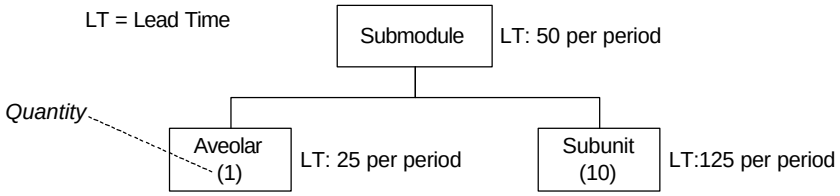



**Figure 68 Example MRP BoM**

The above notation uses the rule that the manufacturing of a composite item cannot be started until all of it's child components are available. Therefore the figure says that a submodule requires 1 aveolar and 10 subunits to be manufactured before its own production can start. Using the information in the above BoM, the following table shows that it would take 5 time periods to produce 50 sub-modules.

| Component | | Period | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Submodule | Planned order receipt | | | | | 50 |
| | Planned order release | | | | 50 | |
| Aveolar | Planned order receipt | | | | 50 | |
| | Planned order release | | 50 | | | |
| Subunit | Planned order receipt | | | | 500 | |
| | Planned order release | 500 | | | | |

**Table 7 Using the information in an MRP BoM**

The *Capacity Requirements Plan (CRP)* determines if the factory has the required capacity to fulfil the manufacturing orders of the MRP. To carry out its job, the CRP

looks at the work centres that make up the factory floor. This is different from the production plan, MPS, and MRP, which consider the factory as a black box. A work centre is a location within a factory where people and/or machines execute the same type of work with the same capacity. CRP uses the work centre master file, work centre agendas, and product route files to translate the MRP manufacturing orders into the hours of work needed at each work centre per period. The CRP schedules work onto work centres to determine if there is enough capacity to fulfil the MRP. The work centre master file gives the setup time, run time, and capacity of each work centre. It also specifies how long it takes to move items between work centres. The work centre agendas say which orders have been released and planned at each work centre. A product route file specifies the sequence of operations required to manufacture one particular product type. For each operation the routing file specifies the work centre to be used, average time taken, and the type of work to be carried out.

# 16 Appendice E - Abréviations

| | |
|---|---|
| **BoM** | **B**ill **o**f **M**aterials |
| **CERN** | **C**onseil **E**uropéen pour la **R**echerche **N**ucléaire |
| **CMS** | **C**ompact **M**uon **S**olenoid |
| **CNRS** | **C**entre **N**ational de la **R**echerche **S**cientifique |
| **DARPA** | **D**efence **A**dvanced **R**esearch **P**rojects **A**gency |
| **ECAL** | **E**lectromagnetic **CAL**orimeter |
| **FSA** | **F**inite **S**tate **A**utomaton |
| **FSTN** | **F**inite **S**tate **T**ransition **N**etwork |
| **LAPP** | **L**aboratoire d'**A**nnecy-le-Vieux de **P**hysique des **P**articules |
| **MRP** | **M**aterial **R**equirements **P**lanning |
| **MRP II** | **M**anufacturing **R**esource **P**lanning |
| **OMT** | **O**bject **M**odelling **T**echnique |
| **SME** | **S**mall to **M**edium sized **E**nterprise |
| **UML** | **U**nified **M**odelling **L**anguage |
| **UWE** | **U**niversity of the **W**est of **E**ngland |
| **Wf** | **W**ork**f**low |