**MASTER EN SCIENCES INFORMATIQUES**

# Spécification et évaluation du comportement des systèmes d'information

Pigneur, Yves

*Award date:*
1984

*Awarding institution:*
Universite de Namur

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

# SPECIFICATION ET EVALUATION DU COMPORTEMENT DES SYSTEMES D'INFORMATION

PAR

YVES PIGNEUR

Dissertation présentée en vue de l'obtention du titre de Docteur en Science (Option Informatique) aux Facultés universitaires Notre-Dame de la Paix - Namur

JURY : Prof. J. Ramaekers, Président du Jury

Prof. F. Bodart, Promoteur

Profs. C. Rolland
M. Léonard
A. van Lamsweerde

décembre 1984

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

## REMERCIEMENTS

Ma reconnaissance va tout spécialement au professeur F. Bodart qui m'a encouragé à entreprendre mais surtout à terminer cette thèse et n'a cessé de me guider tout au long de ce travail. Ce fut un très grand encouragement !

J'adresse mes remerciements aux professeurs C. Rolland de Paris et M. Léonard de Genève pour s'être intéressés à mon travail et pour avoir accepté d'être membres du jury.

Je remercie également les professeurs J. Ramaekers, directeur de l'Institut, et A. van Lamsweerde d'avoir accepté de présider le jury, respectivement d'en être membre.

Je tiens aussi à exprimer mes plus vifs remerciements à A.M. Hennebert et J.M. Leheureux pour leur disponibilité, leur aide, leurs propositions constructives mais aussi pour l'atmosphère amicale et enthousiaste qu'ils ont toujours su entretenir dans le projet.

Je remercie R. Lesuisse et J.L. Hainaut pour leurs suggestions, leurs conseils et leurs paroles d'encouragement. J'exprime également ma profonde gratitute au professeur J. Fichefet qui, directeur de l'Institut à l'époque, m'a proposé d'y travailler comme assistant.

Je n'oublie pas l'accueil chaleureux que m'ont toujours réservé le professeur D. Teichroew et les membres du projet ISDOS/PRISE de Michigan.

J'associe dans l'expression de ma reconnaissance C. Decoux et Y. Vandenkerckhof qui, avec beaucoup de compétence et de gentillesse, ont assuré les travaux de secrétariat, la dactylographie et la réalisation finale de ce document.

J'adresse enfin mes remerciements à Isabelle et à tous ceux qui me sont proches pour leur compréhension, leurs encouragements et leur appui constant.

# SPECIFICATION ET EVALUATION DU COMPORTEMENT DES SYSTEMES D'INFORMATION

Yves Pigneur

# TABLE DES MATIERES

*

* *

# PREAMBULE

## I. MOTIVATION

De nombreuses **méthodologies,** dont on retrouve notamment un échantillon représentatif dans [OLLE, 1982] et [MADDISON, 1983], ont été élaborées et sont actuellement proposées pour **concevoir** les **systèmes d'information,** efficaces et efficients, dont les organisations souhaitent se doter.

Ainsi qu'il avait, par exemple, été défini dans [BODART, 1983] un **système d'information** peut être vu comme une "construction formée d'ensembles

- d'informations, représentations - partielles - des faits qui intéressent l'organisation ;
- de traitements, procédés d'acquisition, de mémorisation, de recherche, de communication et de transformation des informations ;
- mais aussi de ressources humaines, techniques et organisationnelles qui en assurent le fonctionnement".

L'attention portée à ces dernières insiste sur l'adéquation des moyens aux objectifs en réponse à la double contrainte, souvent perdue de vue, d'**efficacité** (atteindre les objectifs) et d'**efficience** (utilisation optimale des ressources).

La **conception** d'un système d'information ne représente en fait que la première phase de son développement, traditionnellement appelée analyse fonctionnelle ou plus récemment analyse conceptuelle. Très schématiquement, l'analyse fonctionnelle a pour but de spécifier, de façon détaillée, une solution fonctionnelle, cohérente et faisable, au départ de besoins exprimés par l'organisation. Sans entrer dans des querelles d'écoles concernant ce qu'il est convenu d'appeler le cycle de vie d'un système d'information, on peut en effet globalement schématiser l'évolution d'un système d'information selon les étapes suivantes :

- l'analyse fonctionnelle qui spécifie fonctionnellement la solution retenue ;
- la conception de mise en oeuvre (design) qui affine et définit la solution adoptée par la prise en compte des caractéristiques logiques des moyens de réalisation humains, techniques et organisationnels ;

- la réalisation et la mise-au-point de la solution en fonction des caractéristiques physiques des matériels, des logiciels et de l'organisation ;
- l'utilisation et la maintenance du système d'information développé.

Enfin, selon le consensus actuel, le contenu et d'ailleurs la spécificité de chacune des **méthodologies** de conception proposées s'expriment en substance par des modèles, des outils logiciels et des méthodes - ou règles - par lesquels elles prétendent assister l'analyste dans son activité de conception.

Toutefois, les expériences accumulées en matière de conception de systèmes d'information, qui s'expriment notamment par la dite "crise du logiciel" ont montré que ces méthodologies s'avèrent encore souvent inadéquates pour garantir l'efficacité et l'efficience des systèmes d'information conçus et développés. Les résultats de récentes conférences [OLLE, 1982], [OLLE, 1983] et groupes d'études [MADDISON, 1983] ont clairement mis en évidence la prolifération de ces méthodologies.

A l'examen, on peut cependant déplorer non seulement le manque de concepts pour classer, harmoniser et unifier les vues en ce domaine mais aussi - et assez manifestement - l'**insuffisante préoccupation** des auteurs à rencontrer le voeux pragmatique des utilisateurs visant l'**adéquation des systèmes d'information** conçus à leurs moyens ou **ressources** humaines, techniques et organisationnelles.

## II. SOLUTION IDA

Face aux déficiences constatées ci-dessus, l'originalité du projet IDA [BODART, 1983] a été de viser à une meilleure cohésion de la méthodologie de conception proposée. Celle-ci se singularise en effet par un effort d'intégration dans un **ensemble cohérent et extensible de modèles, d'outils logiciels et de méthodes de conception.** Elle se caractérise enfin par l'introduction d'une **approche plus expérimentale** dans la conception des systèmes d'information qui permet de tester l'impact de différentes hypothèses de fonctionnement et favorise la perception directe par les utilisateurs de ce qu'ils souhaitent obtenir.

D'une part, sans prétendre régler tous les problèmes posés par cette intégration, les **modèles** élaborés couvrent tous les aspects particuliers des systèmes d'information, retenus par le projet :

- les informations et leur structure ;
- les traitements et leur niveau de décomposition ;
- les conditions d'activation des traitements ;
- les règles d'utilisation et de transformation des informations par les traitements ;
- et enfin, les ressources humaines, techniques et organisationnelles qui exécutent les traitements.

D'autre part, et au départ d'un apport innovateur en logiciel du projet ISDOS (cf. infra point 4 de ce préambule), l'**environnement logiciel** d'IDA fournit le complément indispensable à la mise en oeuvre pratique des modèles, notamment par l'élaboration d'outils logiciels visant à vérifier que la solution spécifiée est complète, cohérente, faisable et conforme aux besoins exprimés par l'utilisateur.

Enfin, il convient de signaler que la démarche méthodologique d'IDA débute l'analyse fonctionnelle proprement dite par une **étude d'opportunité** destinée à objectiver un avant-projet de solution qui reprend notamment, sur base d'évaluations effectuées, une ébauche de la solution retenue, les moyens qu'elle requiert et les efficacités attendues.

## III. NOTRE CONTRIBUTION

Ce travail s'inscrit dans le cadre des préoccupations du projet IDA telles qu'évoquées ci-dessus.

Il apparaît à l'examen que traîter l'intégration mentionnée en matière de modèles, d'outils logiciels et de méthodes passe entre autres mais inévitablement par la prise en compte dans toutes ses implications du comportement des systèmes d'information à concevoir et à spécifier. La **spécification et l'évaluation du comportement** - aussi appelé dynamique - des systèmes d'information constituent en fait l'objet de la présente étude.

La nature même de ce travail impliquait un effort de systématisation, de justification et de rigueur scientifique non pas pour

sacrifier à un excès d'académisme mais avec la préoccupation fondamentale de fournir un instrument de génie logiciel non seulement intellectuellement argumenté mais aussi concrètement achevé et prêt pour une utilisation effective. (*)

Le **premier chapitre** de cette thèse est consacré à la présentation du **modèle adopté pour spécifier fonctionnellement le comportement** des systèmes d'information ainsi qu'à la présentation du **langage** associé. Ce modèle permet de spécifier les quatre aspects suivants du comportement :

- le comportement fonctionnel proprement dit qui correspond principalement aux conditions d'activation ou d'enchaînement des traitements et pour lequel une approche originale "Evénement-Processus" qui intègre une forte composante temporelle a été retenue ;
- les moyens mis en oeuvre ou ressources requises par les traitements ;
- une estimation de la charge et
- les performances attendues du système d'information conçu et à développer.

Ce modèle, via le langage associé, permet également de décomposer, progressivement et de façon contrôlée, les traitements et autorise donc une approche méthodologique par affinements successifs.

Enfin, les différents concepts du modèle font l'objet non seulement d'une description précise, principalement basée sur un "méta-modèle" Entité-Association [CHEN, 1976], [TEICHROEW, 1980] mais aussi d'une représentation sous forme d'instructions d'un langage de spécification ad hoc.

Le **second chapitre** présente l'architecture et les fonctions d'un **outil logiciel dédicacé d'évaluation du comportement** qui a été intégré dans l'environnement logiciel généralisé, adopté par le projet IDA (cf. infra point 4 de ce préambule). (*)

Deux idées sont à la base de l'intégration de cet outil dans cet environnement logiciel de spécification. D'une part, il met en oeuvre une **approche par simulation informatique.** D'autre part, une **technique de génération automatique** (de code) permet de garantir l'absence de divergences entre les spécifications du comportement et le programme de

---

(*) L'outil logiciel d'évaluation par simulation présenté au chapitre 2, après avoir fait l'objet de trois versions prototypes, est actuellement un produit opérationnel intégré dans l'environnement logiciel - commercialisé - d'IDA.

simulation correspondant, une fois la correction du programme de génération vérifiée.

Une partie importante de ce chapitre est également consacrée à la présentation des mesures et résultats statistiques, fournis par l'outil logiciel d'évaluation et susceptibles d'expliquer ou de restituer le comportement du système d'information en cours de conception.

Le **troisième chapitre** concerne l'utilisation de l'environnement logiciel adopté, complété de l'outil d'évaluation par simulation, pour **vérifier les spécifications fonctionnelles du comportement** d'un système d'information, rédigées à l'aide du langage présenté au chapitre 1. Cette vérification revient à garantir que la solution spécifiée est **complète, cohérente et faisable.**

En particulier, le recours à l'outil logiciel d'évaluation par simulation permet d'analyser - qualitativement - le comportement spécifié et donc d'en vérifier la cohérence. Toutefois, son utilisation privilégiée réside essentiellement dans l'assistance qu'il est susceptible d'apporter dans la vérification de la "faisabilité" de la solution fonctionnelle retenue et donc dans l'évaluation des performances du futur système d'information - avec un souci d'ordre quantitatif.

L'intérêt principal de cet outil logiciel de simulation tient dès lors à l'**approche expérimentale** qu'il favorise dans la conception fonctionnelle des systèmes d'information. Il permet en effet d'envisager différentes hypothèses alternatives, de tester leur impact et par conséquent de mieux appréhender ou apprécier le futur comportement du système d'information spécifié. Son utilisation semble donc pouvoir **répondre au souci de concevoir des systèmes d'information efficients,** bien adaptés aux ressources mises en oeuvre, compte tenu de la charge présumée et des performances attendues du futur système. Il constitue ainsi un **outil privilégié de l'étude d'opportunité** qui vise à dégager un avant-projet de solution répondant à l'attente et aux possibilités de l'organisation.

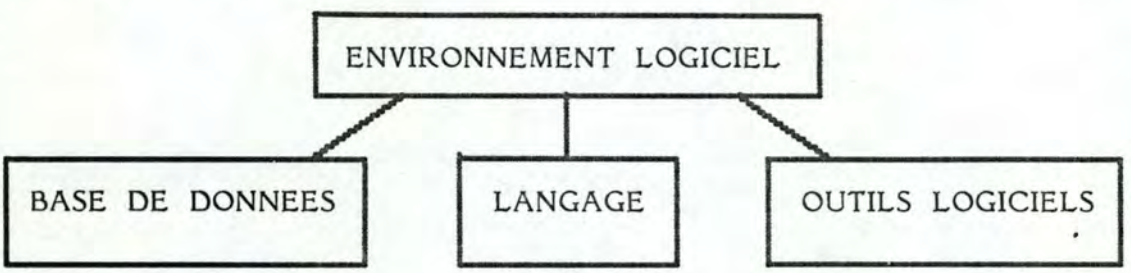
Enfin, une annexe illustre l'application du modèle et de l'environnement logiciel de spécification à un exemple de gestion - simplifiée - de prêts dans un organisme financier.

## IV. ENVIRONNEMENT LOGICIEL DE SPECIFICATION

La brève présentation de l'environnement logiciel de spécification qui suit doit être considérée comme une annexe de ce préambule. Elle est située à cet endroit car cet environnement logiciel constitue un cadre de référence pour le langage (cf. chapitre I.) et les outils logiciels (cf. chapitres II. et III.) développés et/ou utilisés dans cette étude.

L'environnement logiciel généralisé proposé par [YAMAMOTO, 1981] et adopté par les projets ISDOS [TEICHROEW, 1980] et IDA [BODART, 1976-1984] a été retenu comme support logiciel de spécification du comportement des systèmes d'information.

Cet environnement logiciel s'articule autour des trois composants illustrés à la figure suivante :

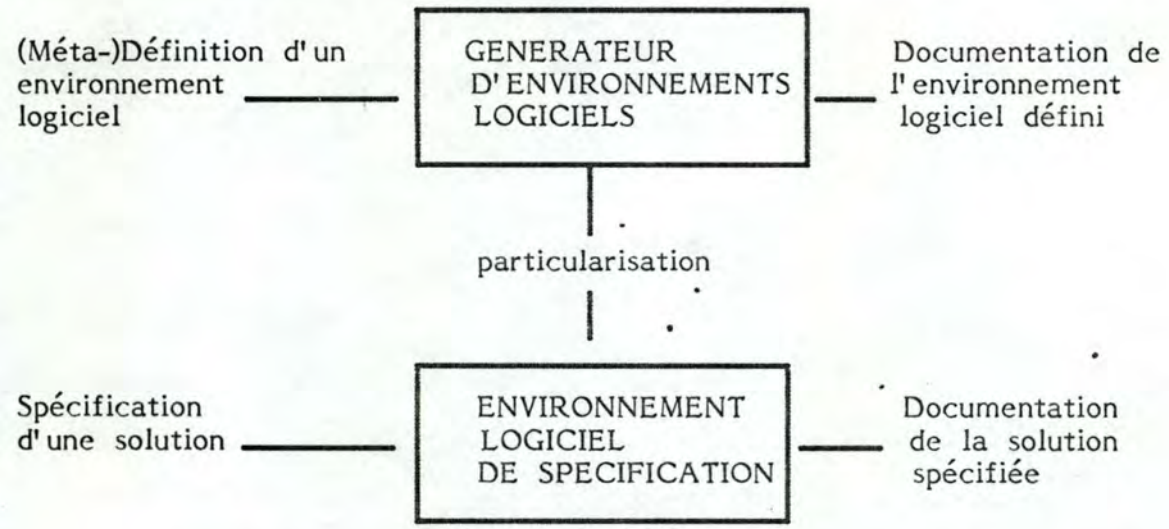



Signalons, avant de préciser chacun de ces trois composants, que cette structure d'environnement logiciel, déjà présente dans le logiciel PSL/PSA du projet ISDOS [TEICHROEW, 1977], se généralise dans la plupart des environnements ou ateliers logiciels [van LAMSWEERDE, 1982] [de DROUAS, 1982], [HUNKE, 1980], [KAMPEN, 1982], [LUCAS, 1979], [OSTERWEIL, 1981].

Toutefois, la particularité de cet environnement-ci de spécification réside dans son indépendance vis-à-vis du modèle de spécification adopté car il a été montré [YAMAMOTO, 1981]

1. qu'il était possible de définir un **"méta-modèle"** de modèles de spécification ;
2. qu'il y avait de grandes ressemblances entre les environnements logiciels pouvant servir de supports à ces modèles ;
3. et donc qu'au départ d'une expression d'un modèle de spécification faite à l'aide de ce méta-modèle, il était possible d'utiliser un **générateur d'environnements logiciels** pour contruire automatiquement - sans reprogrammation - un environnement logiciel dédicacé au modèle retenu.

La figure suivante illustre schématiquement l'approche par génération automatique d'un environnement logiciel dédicacé à un modèle de spécification particulier :




La génération ou construction assistée par ordinateur d'un environnement logiciel de spécification comprend les activités suivantes :

1. définition du modèle de spécification adopté et du langage de spécification associé à l'aide d'un "méta-langage" ;
2. validation de cette définition assistée par l'utilisation de programmes de contrôle ;
3. production - automatique - de la documentation (manuel de référence du langage) ;
4. génération automatique de l'environnement logiciel particularisé au modèle défini.

La "méta-définition" d'un modèle et d'un langage de spécification permet donc d'adapter les trois composants de l'environnement logiciel de spécification :

1. le schéma de la **base de données** ;
2. la syntaxe du **langage de spécification** ;
3. la signification des **outils logiciels** généralisés.

La **base de données** de spécification est l'élément central de l'environnement logiciel. Elle regroupe l'ensemble des spécifications et devient ainsi la source unique de toute la documentation. Les spécifications y sont structurées selon un (méta-)modèle "Entité-Association" [CHEN, 1976]. Le schéma d'une telle base de données est donc adapté à un

modèle de spécification particulier en représentant les concepts du modèle et leurs interactions sous forme de

- types d'entités,
- types ou domaines de valeurs,
- types d'associations et
- types d'attributs.

Le **langage de spécification** permet de rédiger les spécifications dans un formalisme convenu et de les enregistrer dans la base de données par l'intermédiaire d'outils logiciels appropriés. Selon leur forme, on peut distinguer trois types de langages :

- langage "textuel" qui communique les informations sous forme de phrases ou chaînes de caractères ;
- langage "positionnel" dont la syntaxe est déterminée par la position géométrique de ses composants (à la manière des gestionnaires d'écrans) ;
- langage "graphique" qui véhicule les informations à travers des représentations graphiques convenues.

Ces trois types de langages sont utilisables dans l'environnement logiciel ; toutefois, une forme "textuelle" y a été retenue comme moyen privilégié de communication. Il faut cependant signaler que le fait d'avoir clairement dissocié dans le méta-modèle la définition

- du schéma de la base de données (QUOI spécifier) et
- de la syntaxe du langage (COMMENT spécifier)

permet d'utiliser concurremment plusieurs langages différents pour un même modèle de spécification et donc un même schéma de base de données.

Enfin, un langage défini à l'aide du méta-modèle - dont on trouvera une présentation plus complète dans l'introduction du chapitre 1 - est non procédural car il permet de rédiger les spécifications dans un ordre quelconque. Les spécifications sont rédigées en définissant pour des entités :

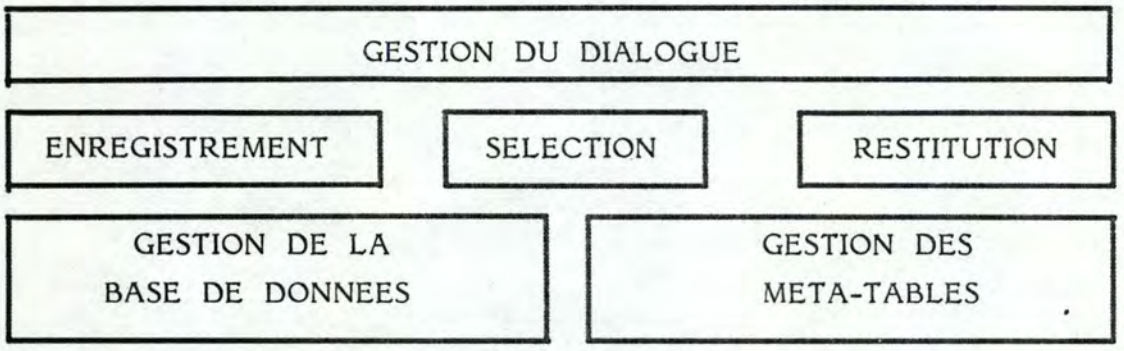
- leur nom et leur type,
- d'éventuels synonymes,
- des associations dans lesquelles elles assument un rôle particulier,
- des attributs et leurs valeurs, en particulier des textes en format libre.

Le méta-modèle permet ainsi de définir les différentes formes syntaxiques associées à chaque concept du modèle de spécification défini.

Les **outils logiciels** sont des programmes informatiques qui assistent le concepteur dans l'élaboration des spécifications. Ces outils permettent :

- d'une part, d'enregistrer dans la base de données des spécifications rédigées à l'aide du langage défini et.
- d'autre part, d'exploiter cette base de données pour effecteur des contrôles et produire des rapports documentaires.

L'architecture générale de ces outils logiciels est illustrée à la figure suivante :




Les outils logiciels d'**enregistrement** acceptent des spécifications rédigées à l'aide du langage, effectuent certains contrôles pour garantir l'intégrité de la base de données et mettent à jour celle-ci.

Les outils de **sélection,** en particulier un langage d'interrogation de haut niveau - plus complètement présenté au chapitre III (cf. III.2.), permettent de sélectionner dans la base de données des informations qui satisfont certains critères.

Les outils de **restitution** produisent des rapports qui restituent la totalité ou une partie des spécifications sous différentes formes plus appropriées à l'exploitation que la forme initiale (instructions du langage).

Les informations extraites et produites par les outils de sélection et de restitution peuvent notamment être utilisées dans un **contexte de validation** et assister l'analyste dans la vérification des spécifications.

L'architecture de cet environnement logiciel, imposée par le caractère non procédural et progressif des spécifications signifie en effet que :

- la base de données peut temporairement contenir des spécifications non vérifiées et donc incomplètes ou incohérentes et que
- la vérification de ces spécifications est effectuée a posteriori et à la demande sur la totalité ou une partie de la base de données.

L'utilisation de ces outils logiciels sera plus complètement présentée au chapitre III dans le cadre de la vérification des spécifications fonctionnelles du comportement des systèmes d'information.

Les rapports produits par les outils de restitution peuvent également servir à des **fins documentaires** et constituer le support d'un dossier de documentation - automatiquement généré - du système spécifié à réaliser.

Enfin, tous ces outils logiciels disponibles dans l'environnement généralisé, mais aussi d'éventuels outils dédicacés car développés dans le cadre d'un modèle particulier (cf. chapitre II.), ont recours aux services d'utilitaires pour **gérer** :

- le **dialogue** entre l'utilisateur et l'environnement logiciel,
- l'accès à la **base de données** de spécification et
- la conduite des programmes en fonction de **(méta-)tables** caractéristiques de l'environnement logiciel dédicacé au modèle de spécification choisi.

*
* *

- CHAPITRE I -

# MODELE ET LANGAGE DE SPECIFICATION DU COMPORTEMENT DES SYSTEMES D'INFORMATION

## I.1. INTRODUCTION

Les concepts du modèle présenté dans ce premier chapitre permettent de spécifier fonctionnellement le comportement - ou dynamique - d'un système d'information. La spécification du comportement comprend 5 parties qui couvrent les aspects suivants :

1. la description du comportement fonctionnel proprement dit,
2. la prise en compte des moyens requis,
3. l'estimation de la charge,
4. l'établissement des performances attendues,
5. la décomposition d'un schéma en sous-schémas.

Le **comportement fonctionnel** proprement dit correspond principalement à l'enchaînement des traitements et peut être spécifié à l'aide des concepts d'**événement** et de **processus.** Il permet de préciser que la survenance d'un événement est causée par un processus et qu'un événement contribue au déclenchement d'un processus.

Les **moyens requis** par les traitements peuvent être pris en compte dans l'expression du comportement à l'aide du concept de **ressource** et des interactions qui permettent de préciser qu'une ressource est requise par un processus.

La **charge** d'un système d'information est estimée en précisant l'**échéancier** des événements externes au système d'information qui fixe le volume et la fréquence des survenances d'événements.

Les **performances attendues** du comportement d'un système d'information sont établies en précisant des délais limites entre les survenances d'événements qui se succèdent dans le temps.

La **décomposition d'un schéma en sous-schémas** revient à appliquer le modèle de spécification adopté pour décrire le comportement interne des processus d'un même type.

Les 5 sections de ce chapitre développent respectivement ces 5 aspects du comportement. Chaque concept du modèle proposé y fait l'objet :

- d'une présentation complète,
- d'une traduction sous forme d'instructions du langage de spécification adopté - brièvement présenté ci-dessous.

Ces 2 aspects apparaissent respectivement sous les titres "modélisation" et "spécification".

**Langage de spécification**

La rédaction des spécifications à l'aide du type de langage prévu dans l'environnement logiciel généralisé (cf. préambule) est **structurée en sections.** Une section comprend une ou plusieurs **instructions** qui caractérisent une ou plusieurs entités [TEICHROEW, 1980] [CHEN, 1976].

La **première instruction** identifie la ou les **entités courantes.** Elle précise le type d'entité et le nom de chaque entité. Si PROCESSUS (cf. I.2.2.), par exemple, est un concept du modèle et donc un type d'entité défini dans la méta-définition (cf. préambule - environnement logiciel adopté), l'instruction suivante commence une nouvelle section :

DEF PROCESSUS enregistrement-commande

Cette instruction signale que la section qui commence spécifie - totalement ou partiellement - une entité de type PROCESSUS dont le nom est "enregistrement-commande" et qui devient courante pour les instructions suivantes.

Les instructions suivantes de la section, dont l'ordre est non significatif (= langage non procédural) caractérisent la ou les entités courantes en spécifiant
- des noms synonymes d'une entité courante,
- des associations entre chaque entité courante et d'autres entités,
- des attributs et leur valeur pour chaque entité courante.

Une instruction peut spécifier un ou plusieurs noms **synonymes** de l'entité courante. L'instruction suivante, par exemple, signale que l'entité courante (enregistrement-commande) peut également être appelée "enreg-comm" :

SYNONYME enreg-comm

Une instruction peut également spécifier une ou plusieurs **associations** entre chaque entité courante et d'autres entités et/ou valeurs constantes. L'instruction suivante, par exemple, précise que l'entité courante qui est un PROCESSUS (enregistrement-commande) requiert l'utilisation d'une RESSOURCE "terminal" pour pouvoir s'exécuter (cf. I.3.) :

REQUIERT 1 terminal

Il faut cependant signaler que cette **même** spécification aurait pu être rédigée de la façon suivante si la RESSOURCE "terminal" avait été courante (DEF RESSOURCE terminal) :

REQUISE PAR enreg-comm AU TAUX DE 1

Ce langage autorise donc généralement qu'une **même** spécification soit rédigée de **plusieurs** manières en fonction du rôle assumé par l'entité courante dans l'association spécifiée (le rôle de requérant ou de requis, par exemple) ; les outils logiciels pouvant automatiquement restituer toutes les formes équivalentes.

Enfin, une instruction peut spécifier un **attribut** et sa valeur pour chaque entité courante. L'instruction suivante, par exemple, spécifie que l'entité courante (le PROCESSUS enregistrement-commande) est un traitement NON interruptible :

INTERRUPTION : REFUSEE

Cela signifie qu'INTERRUPTION est le nom d'un attribut qui peut caractériser les PROCESSUS et dont les valeurs possibles sont REFUSEE et AUTORISEE, par exemple. Mais, la valeur de certains attributs peut également être un **texte** descriptif en format libre. L'instruction suivante, par exemple, complète la spécification de l'entité courante (enregistrement-commande) en décrivant ses règles de traitement sous forme d'un texte en format libre :

PROCEDURE : 1. vérifier l'identité du client
2. créer le client si le client est inconnu
3. vérifier le corps de la commande
4. enregistrer les lignes valides de la commande

Un **texte** est considéré comme une description en format libre par l'environnement logiciel généralisé. Toutefois, il peut être nécessaire, dans certains cas, de formaliser ces descriptions, c'est-à-dire d'utiliser une **syntaxe** - dite **imbriquée** [KONSYNSKI, 1975] - convenue pour les rédiger. Cette formalisation permettra à certains outils logiciels dédicacés d'exploiter ces descriptions dans le cadre de fonctions de génération ou de vérification automatique, par exemple.

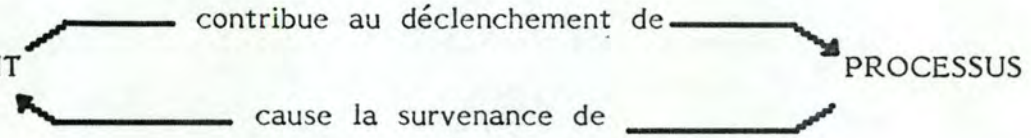
Ainsi, en plus de la méta-définition du langage et dans la mesure où un outil logiciel a été intégré dans l'environnement logiciel généralisé pour évaluer le comportement (cf. chapitre II), une expression formelle du prédicat des conditions de déclenchement (cf. I.2.5.) a également été définie.

Enfin, une section se termine quand une autre section commence ou quand il n'y a plus d'instruction.

## I.2. MODELISATION ET SPECIFICATION DU COMPORTEMENT FONCTIONNEL

Le comportement fonctionnel d'un système d'information est décrit à l'aide d'un modèle causal dans lequel un **événement** est un stimulus auquel le système d'information réagit par le **déclenchement** de **processus** ou traitements qui peuvent à leur tour causer la **survenance** d'autres événements.

La figure suivante illustre les interactions entre les 2 concepts de base du modèle proposé pour décrire le comportement fonctionnel d'un système d'information :




Après avoir proposé une vue synthétique des approches "événement-processus" que l'on retrouve dans la littérature, cette section présentera les 2 concepts de **processus** et d'**événement** ainsi que les interactions de **survenance** et de **déclenchement** qui les relient pour exprimer le comportement souhaité d'un système d'information. Cette section se terminera par la mise en évidence d'une relation d'ordre chronologique entre les événements et les processus.

### I.2.1. L'approche EVENEMENT-PROCESSUS dans la littérature

La modélisation du comportement d'un système d'information à l'aide des concepts d'**événement** et de **processus** a fait l'objet de nombreuses propositions qui diffèrent cependant, et parfois considérablement, au niveau des définitions adoptées.

Parmi les différents travaux qui proposent une approche **événement-processus** pour exprimer le comportement d'un système d'information, 2 grandes tendances émergent selon que leur modèle considère qu'un événement

- correspond à un changement d'état d'informations référencées dans leur schéma conceptuel des données (approche par les données) ou
- est provoqué à l'occasion d'un traitement (approche par les traitements).

Nous avons sélectionné quelques modèles représentatifs de ces 2 tendances en insistant chaque fois sur leurs caractéristiques originales et en signalant ce que nous retenions de ces propositions.

Les modèles que nous avons sélectionnés l'ont été car :

1. ils ont pour objectif de décrire le comportement d'un système d'information à un niveau fonctionnel ou conceptuel ;
2. ils proposent une approche explicitement basée sur les concepts d'événement et de processus.

Cela signifie que d'autres modèles ont été écartés car

1. ils utilisent les concepts d'événement et de processus dans d'autres contextes (simulation, système d'exploitation, langage de programmation...) [RIDDLE, 1978] [SHAW, 1980] [REUVENI, 1980] [REED, 1979]
2. ils expriment le comportement d'un système d'information à l'aide d'autres concepts [ROSS, 1977] [BELL, 1977] [HARRISON, 1981] [ALFORD, 1981] [BRODIE, 1982] [DAVIS, 1978] [WASSERMAN, 1982].

Approche par les données

1. Dans [BUKENKO, 1979], un événement représente "une décision ou une observation, survenant à un moment donné, du début, de la fin ou de la survenance d'une association (= information) définie dans le modèle (conceptuel) des données. Un événement est toujours lié à un repère temporel : sa date de survenance ou d'observation".

De façon fort similaire dans [FALKENBERG, 1975-1979] [BREUTMANN, 1979] "toute association (= information) a un début et une fin considérés comme événements sur l'axe du temps" mais de plus "à un événement correspondant à l'insertion, la suppression ou la modification d'une information d'un type donné est associée une procédure ou règles sémantiques à vérifier lorsque l'événement survient".

Davantage que le rôle de l'événement dans la description du comportement d'un système d'information, - encore peu explicitement souligné, nous retiendrons de ces modèles leur prise en compte explicite du temps dans la modélisation d'un système d'information.

2. Dans [BENCI, 1976] par contre, dont la proposition apparaît comme précursive au développement d'autres approches de cette tendance par des données [BRACCHI, 1979] [ROLLAND, 1980-1981] [LEONARD, 1978], "un événement représente une décision qui déclenche l'exécution d'opérations sur les données (...). La survenance d'un événement peut être provoquée par l'organisation (événement initial) ou correspondre à un état particulier de la base. Dans ce dernier cas, on distingue les événements qui déclenchent des opérations (événements internes) et ceux qui sont connus de l'organisation (événements résultats)".
Le rôle central de l'événement dans la modélisation du comportement est donc clairement affirmé par l'intermédiaire des règles d'évolution qui définissent

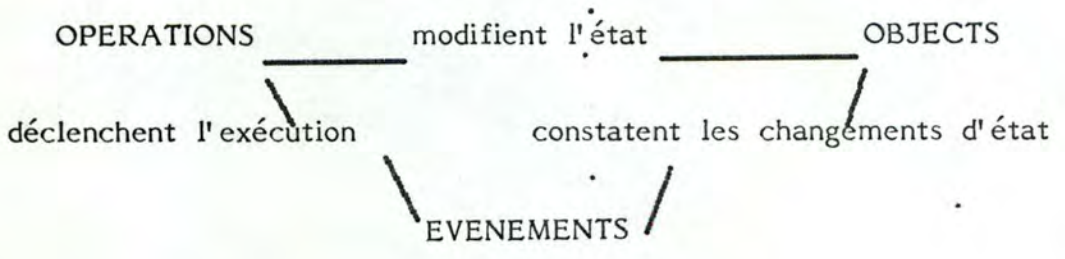
   - les opérations à exécuter sur les données lorsque certains événements surviennent ainsi que
   - les contraintes d'intégrité qui doivent être vérifiées pour que les opérations puissent s'exécuter.

   Nous retiendrons essentiellement de ce modèle

   - le rôle de l'événement qui déclenche des opérations,
   - la notion de paramètre d'événement qui fait référence à certaines données du schéma conceptuel et qui caractérise le changement d'état d'un système d'information,
   - le concept de contrainte d'intégrité qui représente une condition d'application d'un traitement.

3. Dans [ROLLAND, 1980-1981] [FOUCAUT, 1982] - projet REMORA, dont la présentation très systématique met clairement en évidence les 3 composantes que l'on retrouve dans les modèles représentatifs de cette approche par les données, un "événement constate un changement d'état d'un objet (= information) ; il déclenche l'exécution d'opérations qui modifient l'état d'autres objets et provoquent ainsi parfois de nouveaux événements".

   Le schéma suivant illustre bien cette approche par les données :



L'évolution des événements et des processus est alors définie par :

- les dates de survenances des événements,
- les dates de déclenchement et de terminaison des opérations,
- les associations qui constatent le changement d'état d'un objet par un événement et qui sont caractérisées par un prédicat qui précise la nature du changement,
- les associations de déclenchement d'une opération par un événement et qui sont caractérisées par d'éventuels
  . facteurs de déclenchement (combien d'opérations déclencher) et
  . conditions de déclenchement (quelles opérations déclencher).

Nous retiendrons surtout de ce modèle :

- la formalisation systématique et rigoureurse du rôle de l'événement et des modalités de déclenchement des opérations par les événements,
- les propriétés temporelles (dates) qui caractérisent le passage d'un état à l'autre d'une opération ou d'un événement.

4. Les derniers modèles que nous avons considérés comme représentatifs de cette approche par les données insistent encore davantage sur les conditions de survenance d'un événement.

Ainsi dans [GOLDMAN, 1980] "Quand une condition sur les objets manipulés par les processus est vérifiée, un processus est exécuté" ou encore dans [HSU, 1979] "un événement survient instantanément lorsqu'un prédicat sur certains objets du monde réel devient vrai".

De façon encore plus explicite dans [GUSTAFSON, 1982] : "la propriété de base d'un événement est qu'il représente un phénomène qui survient instantanément et n'a aucun attribut qui varie avec le temps" (...) "chaque événement est lié à un événement temporel et a une condition de survenance associée". La condition de survenance précise quelles sont les conditions qui doivent être vérifiées dans le système pour qu'un événement d'un type donné survienne et produise ses effets, c'est-à-dire déclenche l'exécution d'opérations sur les données.

Nous retiendrons principalement de ces derniers modèles :

- la notion de condition de survenance qui peut éventuellement décrire ou constater le changement d'état de plusieurs objets ou même correspondre à la survenance de plusieurs événements élémentaires.

L'**intégration** des modélisations du **comportement** et des **données** impose pratiquement une description préalable des données avant de pouvoir exprimer le comportement. Dans certains cas, si on impose qu'un événement corresponde à un changement d'état d'une seule donnée élémentaire, la description des données doit même tenir compte de l'expression du comportement. On pourrait ainsi être amené à devoir considérer dans un schéma conceptuel une commande-client comme autant d'objets que d'états qu'elle est susceptible de prendre en fonction des traitements.

Dans une approche par les traitements par contre, le comportement n'est pas explicitement lié à la structuration des données mais, les événements sont directement provoqués ou causés par les traitements.

Approche par les traitements

1. Dans [SERNADAS, 1980], "un système d'information est modélisé par un réseau de processus concurrents et autonomes qui échangent des messages entre eux et avec l'environnement (...). Parmi ces messages qui constituent la base de données temporelle du système, on distingue les événements qui sont les messages qui apparaissent instantanément dans le système (...). Les événements constituent le mécanisme de déclenchement des processus".

   Nous retiendrons de ce modèle :

   - la modélisation du comportement faite, seulement, à l'aide de 2 concepts - événement et processus - sans référence explicite à la structuration des données.

2. Les modèles suivants utilisent un formalisme dérivé des réseaux de Petri [PETERSON, 1977] [NUTT, 1974] [BRAMS, 1983] pour exprimer le comportement d'un système d'information, en attachant une sémantique convenue aux concepts de place et de transition.

   Ainsi dans [ZISMAN, 1978] et de façon quasi-équivalente dans [BARROW, 1981] "les réseaux de Petri sont utilisés pour modéliser les relations temporelles entre les événements et les paires condition-action pour représenter la connaissance associée à chaque événement".
   Les événements correspondent
   soit à l'arrivée d'un jeton dans une place (une condition devient vraie) ;
   soit au franchissement d'une transition (une action est exécutée).

D'une manière assez voisine pour [DE ANTONELLIS, 1981] et [ATZENI, 1982], les réseaux de Petri constituent un formalisme approprié pour représenter les événements : un graphe avec des places et des transitions liées entre elles par des arcs pour représenter respectivement des conditions, des actions et des relations de précédence entre elles. Dans ce contexte, "un événement exprime un changement de condition entre des conditions vérifiées avant (pré-conditions) et des conditions vérifiées après (post-conditions). Un événement est décrit par ses pré-conditions et les actions qui produisent le changement (les post-conditions sont la conséquence de ces changements)".

De plus, une typologie des conditions est proposée dans laquelle une condition peut correspondre à un changement provoqué par

- une action interne au schéma,
- une action externe au schéma,
- un calendrier.

Enfin, les préconditions et les actions associées à un événement peuvent être élémentaires ou composées - en utilisant des opérateurs booléens - pour exprimer des situations classiques de séquence, de parallélisme, d'exclusion et de synchronisation des traitements.

Nous retiendrons essentiellement de ces modèles basés sur les réseaux de Petri (on aurait également pu citer [ELLIS, 1979-1980])

- leur vue "pré" et "post" conditions à l'activation des traitements.

3. Enfin, la proposition MERISE, [TARDIEU, 1983] [HECKENROTH, 1980] n'utilise pas directement les réseaux de Petri pour modéliser le comportement - qu'ils appelent ciménatique - d'un système d'information. Par contre, il a été montré qu'il était possible d'établir une correspondance entre leur modèle et les réseaux de Petri.

   Dans cette proposition,

   - une opération représente un traitement effectué par le système d'information en réaction à la survenance d'événements ;
   - un événement est un stimulus externe ou interne. Dans ce dernier cas, il est produit à l'intérieur du système d'information lors d'une opération. Un événement est repéré par un message comprenant des informations sur le sytème lors du changement d'état, en particulier le moment et le lieu de l'événement. La production d'un événement par une opération peut être conditionnée par une règle d'émission ou proposition logique sur le contenu de la base de données ;
   - une synchronisation représente une pré-condition pour l'activation d'une opération à partir de plusieurs événements. Un prédicat précise

comment les événements participent à la synchronisation et certaines contraintes portant sur les propriétés temporelles des événements participants peuvent être ajoutées.

Nous retiendrons de ce modèle très proche de celui que nous proposons,

- la présentation très systématique des modalités de contribution des événements au déclenchement des processus.

Par rapport à ces différentes propositions, notre modèle

1. privilégie une **approche par les traitements,** basée sur les concepts d'événement et de processus. Cette approche assure l'**autonomie** de la modélisation du comportement par rapport à la description des données tout en attribuant aux processus un **rôle intégrateur** puisqu'
   - ils sont déclenchés par et causent la survenance des événements (vue Processus-Evénements),
   - ils utilisent des données (vue Processus-Données),
   - ils requièrent des ressources (vue Processus-Ressources) ;

2. décrit très systématiquement le rôle de l'événement et les **modalités de déclenchement** des processus par les événements ;

3. prend en compte très explicitement le **temps** dans la modélisation du comportement (délai, calendrier, échéancier ...) ;

4. permet de décrire aussi bien un comportement global, lorsque les traitements sont encore fort **agrégés,** qu'un comportement détaillé quand les traitements ont fait l'objet d'une décomposition plus **désagrégée.**

### I.2.2. PROCESSUS

**Modélisation**

Un **processus** représente l'**exécution d'une procédure** qui correspond à un traitement à effectuer dans un système d'information. C'est donc une entité dynamique qui apparaît à chaque exécution de la procédure.

Le **nom** du processus est un identifiant qui devrait indiquer la nature du traitement. La **procédure** d'un processus est l'ensemble des règles à suivre ou des actions à entreprendre pour que le processus accomplisse la fonction qui lui est assignée dans le système d'information. L'exécution d'une procédure se traduit généralement par des opérations sur certaines informations véhiculées et/ou enregistrées dans le système d'information.

Dans la mesure où l'on a opté pour l'autonomie des spécifications du comportement par rapport aux spécifications des données et de leur utilisation par les processus, la procédure est vue comme une "boîte noire" au niveau du comportement. Seuls ses effets "externes" (cf. survenance I.2.4.) seront donc pris en considération. Elle fait cependant l'objet d'une description - non formalisée - plus ou moins complète car celle-ci pourrait servir dans un éventuel affinement des spécifications par une décomposition en sous-processus (cf. I.6.).

La **date** de **déclenchement** d'un processus est une propriété implicite qui précise l'instant où un mécanisme de déclenchement a provoqué la création du processus (cf. I.2.5.). La **date de terminaison** est une autre propriété temporelle qui correspond au moment où le processus a accompli toutes les actions prévues dans sa procédure. Entre ces deux moments, le processus est en cours d'exécution ; avant son déclenchement, il était inexistant et après sa terminaison, il a produit tous ses effets et n'a plus aucune incidence sur l'état du système d'information.

Tout processus appartient à la classe qui regroupe les processus de même nature définis par un nom identique et caractérisés par la même procédure. A un même moment, plusieurs processus appartenant à la même classe, c'est-à-dire exécutant la même procédure peuvent être en cours d'exécution à différents stades d'avancement.

Exemples 1.1

> Les "enregistrements d'une commande" sont des processus qui peuvent être en cours d'exécution dans le système d'information associé à une gestion de commandes et opèrent des transformations similaires sur l'état du système selon les mêmes règles de traitement.
> Les "mises à jour du stock" et les "archivages d'une commande" constituent d'autres processus dans le système d'information.

**Spécification**

Une classe de **processus**

| PROCESSUS |
| --- |
| nom<br>procédure |

est définie par :
- un **nom** qui correspond à celui des processus de la classe,
- une description de la **procédure** associée.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide des instructions suivantes :

```
DEF PROCESSUS    nom-de-processus
  PROCEDURE :
    règles de la procédure exprimées sous forme d'un texte libre
```

### I.2.3. EVENEMENT

**Modélisation**

Un **événement** représente un **changement d'état** qui survient à un moment donné de l'évolution d'un système d'information et qui correspond à un **stimulus** auquel ce système doit réagir.
Un événement est **interne** s'il survient à l'occasion d'un traitement du système d'information. Il résulte généralement de la transformation d'une information véhiculée et/ou enregistrée dans le système d'information. Un événement est **externe** s'il est provoqué par l'environnement du système d'information. Un événement externe est **temporel** s'il est causé par le seul écoulement du temps.

Le **nom** d'un événement est un identifiant qui devrait indiquer la nature du changement d'état.
Un **attribut d'événement** est une propriété facultative qui précise la nature du changement d'état. Un attribut porte un nom et prend des valeurs qui affecteront la réaction du système d'information à la survenance de l'événement. Un tel attribut fait, généralement, référence à une ou plusieurs informations véhiculées et/ou enregistrées par le système d'information dont les valeurs au moment du changement d'état ont une incidence sur le comportement du système d'information.

La **date** de **survenance** d'un événement est une propriété implicite qui précise l'instant où le changement d'état est survenu.

Tout événement appartient à la classe qui regroupe les événements de même nature, définis par un nom identique et caractérisés par les mêmes attributs.

Exemple 1.2

Chaque "rupture de stock" est un événement - interne - qui survient dans l'évolution d'un système d'information associé à une gestion de commandes. Le "montant de la rupture" est un attribut dont la valeur a une incidence sur les effets que la "rupture de stock" peut avoir sur le comportement dynamique du système d'information.

Chaque "arrivée de commande" est un événement - externe - qui est provoqué par l'environnement du système d'information. Le "degré d'urgence" est un attribut de l'événement qui correspond à une information, associée à la commande reçue, en fonction de laquelle se fera la réaction du système d'information.

Chaque "fin du jour" constitue un événement - temporel - qui survient tous les jours à la même heure.

**Spécification**

Une classe d'**événements**

| EVENEMENT |
| --- |
| nom<br>attribut(s) |

est définie par :

- un **nom** qui est celui des événements de la classe ;
- les éventuels **attributs d'événements** de la classe. Un attribut est défini par un nom et l'ensemble des valeurs qu'il peut prendre et en fonction desquelles se fait la réaction du système.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide des instructions suivantes :

| | |
| --- | --- |
| DEF EVENEMENT | nom-d'événement |
| ATTRIBUT | nom-d'attribut |

La classe des événements "arrivée de commande" de l'exemple 1.2. serait spécifiée comme suit :

| | |
| --- | --- |
| DEF EVENEMENT | arrivée-de-commande |
| ATTRIBUT | degré-d'urgence |
| DEF DOMAINE-DE-VALEURS | degré-d'urgence |
| VALEURS | normal, bas, haut |

## I.2.4 SURVENANCE

**Modélisation**

Les événements internes portent à la connaissance du sytème d'information des changements d'état qui surviennent ou sont causés lors de l'exécution des processus. Ils correspondent parmi les transformations d'informations véhiculées et/ou enregistrées dans le système d'information à celles qui ont une incidence sur son comportement. Une association de survenance représente la **survenance d'un événement causée par un processus.**

Toute association de survenance appartient à la classe des survenances qui relient les processus d'une même classe aux événements d'une même classe.

Exemple 1.3

> Les "ruptures de stock" sont des événements dont la survenance est causée par les processus "mises à jour du stock" dans un système d'information associé à une gestion de commandes.

**Modalités de survenance**

La survenance d'un événement est toujours causée par un seul processus même si la survenance des événements d'une même classe peut être causée par les processus d'une même classe ou de classes différentes. Par contre, un processus peut causer la survenance de plusieurs événements d'une même classe ou de classes différentes. Dans ce cas, une classe de survenances peut donc faire l'objet d'une contrainte de **cardinalité,** qui précise le nombre d'événements dont un même processus cause la survenance.

De plus, la survenance de certains événements peut ne pas être systématiquement causée par un processus. On doit donc pouvoir spécifier dans quelle condition un processus cause la survenance de certains événements. Un prédicat spécifie alors cette **condition de survenance** en fonction de la logique du problème et donc de la spécification complète du système d'information (traitements et données). Pour les mêmes raisons que celles exposées dans la présentation de la procédure (cf. I.2.2.), un tel prédicat est considéré comme une "boîte noire" au niveau du comportement et la spécification ne fait que mentionner son existence. Toutefois, dans une optique d'évaluation du

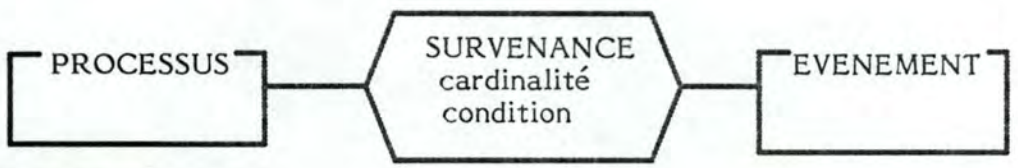
comportement une condition de survenance est également caractérisée par une propriété qui précise la **probabilité** que son prédicat soit vérifié.

Exemple 1.4.

> Chaque processus de "préparation d'une réquisition" cause la survenance d'autant d'événements "livraison à effectuer" que le nombre de commandes groupées ayant fait l'objet de la réquisition.
>
> Un processus d'"enregistrement de commande" cause la survenance d'un événement "commande à traiter" si la "commande est valide" **ou** d'un événement "commande refusée" si la "commande est invalide".

**Spécification**

Une classe de **survenances**




est définie par :

- le nom de la classe des **processsus** qui causent les survenances ;
- le nom de la classe des **événements** dont la survenance est causée ;
- les **modalités** (cardinalité et condition) **de survenance** associées. Si aucune modalité n'est spécifiée, on suppose que chaque processus cause systématiquement la survenance d'**un seul** événement. La cardinalité peut être exprimée sous la forme d'un nombre ou d'une distribution de probabilité.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide d'une des deux formes syntaxiques suivantes :

a. DEF PROCESSUS nom-de-processus
CAUSE [cardinalité] nom-d'événement [SI nom-de-condition]

b. DEF EVENEMENT nom-d'événement
CAUSE [cardinalité FOIS] PAR nom-de-processus [SI nom-de-condition]

où une condition de survenance est rédigée de la façon suivante :

DEF CONDITION nom-de-condition
PREDICAT :
VRAIE-DANS (probabilité)

Les classes de survenances de l'exemple 1.4 peuvent être spécifiées de la façon suivante :

```
DEF PROCESSUS   préparation-d'une-réquisition
    CAUSE       un-certain-nombre    réquisition-à-effectuer

DEF PROCESSUS   enregistrement-de-commande
    CAUSE       1 commande-à-traiter SI commande-est-valide,
                1 commande-refusée SI commande-est-invalide
```

La spécification des classes de survenances ayant trait à une même classe de processus constitue une **"post-condition"** dynamique de ces processus puisqu'elle donne une vue partielle de l'état dans lequel le système d'information se trouve lorsqu'un tel processus se termine.

### I.2.5 DECLENCHEMENT

**Modélisation**

Les événements sont des stimuli auxquels le système d'information réagit par le déclenchement de processus. Une association de **déclenchement** représente la contribution d'un événement au déclenchement d'un processus.

Toute association de déclenchement appartient à la classe des déclenchements qui relient les événements - contribuants - d'une même classe aux processus - déclenchés - d'une même classe.

Exemple 1.5.

Chaque événement "arrivée d'une commande" déclenche un processus d'"enregistrement d'une commande".
Si les commandes du jour sont archivées en fin de journée, un processus d'"archivage d'une commande" est déclenché lorsqu'un événement "arrivée d'une commande" et un autre "fin du jour" sont survenus.

**Modalités de déclenchement**

L'existence d'une association de déclenchement peut être soumise à certaines contraintes. Une contrainte est exprimée sous la forme de prédicats qui doivent être vérifiés pour que l'événement contribue au déclenchement d'un processus. Un prédicat est une expression booléenne dont les opérateurs sont les opérateurs booléens (ET, OU, NON) et les opérandes des expressions de comparaison. Une expression de comparaison est une expression qui délivre une valeur booléenne (VRAI ou FAUX) et dans laquelle les opérateurs sont les opérateurs de comparaison (< , > , = , <= , >= , < >) et les opérandes des caractéristiques des associations de déclenchement. Ces caractéristiques peuvent

être :

- les attributs des événements contribuants ;
- le délai entre la survenance des événements contribuants et le déclenchement des processus déclenchés ;
- le nombre maximum de déclenchements auxquels un même événement **peut** contribuer ainsi que le nombre d'événements qui **doivent** contribuer au déclenchement d'un même processus.

En l'absence de parenthèse, l'évaluation d'une telle expression booléenne se fait selon la hiérarchie habituellement convenue NON, ET, OU.

Les associations d'une même classe sont caractérisées par les mêmes prédicats qui traduisent une même modalité de déclenchement. Lorsqu'une contrainte porte sur les associations de déclenchement d'une seule classe, le prédicat ne fait intervenir que les caractéristiques propres aux associations de la classe. Par contre, un prédicat peut faire intervenir les caractéristiques des associations de plusieurs classes pour traduire une contrainte "jointe" portant sur les associations de différentes classes.

Exemple 1.6

> Si on a décidé de n'archiver que les commandes signées, un événement "arrivée de commande" ne contribue au déclenchement d'un processus "archivage d'une commande" que si la valeur de l'attribut d'événement "présence signature" est vraie.
>
> Le délai entre la survenance d'une "arrivée de commande" et le déclenchement de l'"archivage de la commande" doit toujours être supérieur au délai entre la survenance de la "fin du jour" et le déclenchement de l'"archivage de la commande" pour garantir qu'il ne s'agit pas d'une "fin du jour" antérieure à l'"arrivée de la commande".
>
> L'"arrivée d'une commande" ne peut contribuer au déclenchement que d'un seul "enregistrement" et d'un seul "archivage d'une commande". Par contre, une même "fin du jour" peut contribuer au déclenchement de plusieurs "archivages d'une commande".
> Enfin un "enregistrement d'une commande" est déclenché par une seule "arrivée d'une commande" et un "archivage d'une commande" est déclenché par une seule "arrivée d'une commande" **et** une seule "fin du jour".

On peut dégager une typologie de ces modalités de déclenchement en fonction du type de caractéristique sur lequel porte la contrainte :

1. valeur des attributs d'événements contribuants,
2. délai de contribution,
3. nombre de contributions.

Nous allons expliciter ces trois types de contraintes, en examinant chaque fois ce qu'elle signifie, lorsque

A. un événement contribue au déclenchement d'un processus,

B. un événement contribue au déclenchement de plusieurs processus,

C. plusieurs événements contribuent au déclenchement d'un processus.

### 1. Modalités qui portent sur la valeur des attributs d'événements

Lorsqu'une contrainte (de déclenchement) porte sur les attributs des événements contribuants, elle peut traduire les comportements suivants :

A/B : Un événement d'une classe $E_1$ peut contribuer au déclenchement de processus d'une même classe (ou de différentes classes $P_1$, $P_2$, ..., $P_N$)
ssi
une expression de comparaison dans laquelle intervient un attribut $A_1$ de l'événement est vraie.

Ce type de contrainte traduit un phénomène de décision (cf. I.2.6.) dans le comportement d'un système d'information.

Exemple 1.7

L'événement "arrivée d'une commande" peut contribuer au déclenchement d'un processus "enregistrement d'une commande" si l'attribut "présence signature" a la valeur vraie ou au contraire contribuer au déclenchement d'un processus "refus d'une commande" si le même attribut à la valeur fausse.

C : Des événements de différentes classes $E_1$, $E_2$, ..., $E_N$ peuvent contribuer au déclenchement d'un même processus d'une classe $P_1$
ssi
une expression de comparaison dans laquelle interviennent des attributs $A_1$, ..., $A_N$ de ces événements est vraie.

Ce type de contrainte permet d'exprimer un phénomène de synchronisation (cf. I.2.6.) dans le comportement d'un système d'information car cette contrainte permet de choisir la combinaison autorisée des événements des classes $E_1$, ..., $E_N$ pouvant effectivement contribuer au déclenchement d'un processus de la classe $P_1$.

Exemple 1.8

Un événement "entête de commande vérifiée" et un autre "corps de commande vérifié" pour lesquels la valeur de leur attribut "n° de commande" est identique contribuent au déclenchement d'un processus "traitement d'une commande".

## 2. Modalités qui portent sur le délai de contribution

Lorsqu'une contrainte de déclenchement porte sur le délai entre la survenance des événements contribuants et le déclenchement des processus, elle peut traduire les comportements suivants :

A. Un événement d'une classe $E_1$ peut contribuer au déclenchement de processus d'une classe $P_1$
ssi
une expression de comparaison dans laquelle intervient le délai entre la survenance d'un événement $E_1$ et le déclenchement d'un processus $P_1$ est vraie.

Lorsque ce délai doit être inférieur à une certaine durée, il représente une période de **remanence** au-delà de laquelle un événement $E_1$ ne peut plus contribuer au déclenchement d'un processus $P_1$.
Un délai nul signifie qu'un événement $E_2$ ne peut contribuer au déclenchement d'un processus $P_1$ qu'au moment de sa survenance. En d'autres termes, ses effets doivent être immédiats, comme si le système ne mémorisait pas sa survenance.
Enfin, lorsque le délai doit être supérieur à une durée donnée, il correspond à une période de **temporisation** entre la survenance d'un événement $E_1$ et son éventuelle contribution au déclenchement d'un processus $P_1$.

B. Un événement d'une classe $E_1$ peut contribuer au déclenchement de processus de classes $P_1$, ..., $P_N$
ssi
une expression de comparaison dans laquelle interviennent les délais entre la survenance d'un événement $E_1$ et le déclenchement de processus $P_1$, ..., $P_N$ est vraie.

Ce type de contrainte permet, principalement, de préciser qu'un même événement $E_1$ ne peut plus contribuer au déclenchement d'un processus $P_i$ si il a, préalablement, contribué à celui d'un processus $P_j$.

Exemple 1.9

Si un événement "autorisation d'enregistrer" peut contribuer au déclenchement d'un "enregistrement" à condition qu'il n'ait pas déjà contribué au déclenchement d'un processus de "clôture d'autorisation", on signalera que le délai entre la survenance d'une "autorisation d'enregistrer" et le déclenchement d'un "enregistrement" est inférieur à celui entre la survenance de la même "autorisation" et le déclenchement d'une "clôture".

C. Des événements de différentes classes $E_1$, ..., $E_N$ peuvent contribuer au déclenchement d'un même processus d'une classe $P_1$ ssi une expression de comparaison dans laquelle interviennent les délais entre la survenance des événements $E_1$, ..., $E_N$ et le déclenchement d'un processus $P_1$ est vraie.

Ce type de contrainte permet, principalement, de préciser que certains événements doivent être plus récents que d'autres pour pouvoir contribuer au déclenchement d'un même processus.

Exemple 1.10

Un événement "fin du jour" et un autre "arrivée d'une commande" contribuent au déclenchement d'un processus "archivage de la commande" à condition que le délai entre la survenance de la "fin du jour" et le déclenchement de "l'archivage de la commande" soit inférieur à celui compris entre la survenance de l'"arrivée de la commande" et le déclenchement du même "archivage ...". Cette contrainte garantit simplement que la "fin du jour" précédent n'a plus aucune influence sur les commandes du jour.

**3. Modalités qui portent sur le nombre de contributions**

Un événement n'entraîne, généralement, qu'un nombre limité d'effets et un processus n'est déclenché que par un certain nombre d'événements. Lorsqu'une contrainte porte sur le nombre de déclenchements auxquels un même événement peut contribuer ou sur le nombre d'événements qui doivent contribuer au déclenchement d'un processus, elle peut traduire les comportements suivants :

A.1. Une contrainte peut préciser le nombre de déclenchements de processus d'une classe $P_1$ auxquels un même événement $E_1$ peut contribuer. Par défaut, un événement peut contribuer au déclenchement d'un seul processus d'une classe $P_1$. Puisque cette remarque est valable pour toutes

les classes de déclenchements dans lesquelles intervient cet événement, cela permet de prendre en considération des phénomènes de parallélisme (cf. I.2.6.) dans les traitements.

Exemple 1.11

L'événement "arrivée d'une commande" contribue au déclenchement d'un "enregistrement d'une commande" et d'un "archivage de commande" (règle par défaut) qui peuvent donc s'exécuter en parallèle.
Par contre, un événement "fin du jour" peut contribuer au déclenchement de plusieurs processus "archivage".

A.2. Une contrainte peut préciser le nombre d'événements d'une même classe $E_1$ qui doivent contribuer au déclenchement d'un même processus $P_1$. Par défaut, il suffit d'un seul événement pour chaque classe de déclenchements dans laquelle intervient ce processus pour déclencher celui-ci.

Exemple 1.12

Un événement "arrivée d'une commande" et un autre "fin du jour" contribuent au déclenchement d'un "archivage ..." (règle par défaut), mais il faut 10 événements "commande enregistrée" pour déclencher un seul "parcours en magasin".

B. Une contrainte peut préciser le nombre de déclenchements de processus de classe $P_1$ ... $P_N$ auxquels un même événement $E_1$ peut contribuer. Une telle contrainte permet de prendre en considération un comportement similaire à celui d'un réseau de Petri où un jeton ne peut contribuer à l'activation d'une seule transition parmi celles qui peuvent l'être par les jetons d'une même place. Il faut, cependant, faire remarquer que, sans autre contrainte ou règle, ce type de contrainte peut entraîner un conflit ou comportement indéterminé puisqu'en cas de survenance d'un tel événement, le choix du type de processus à déclencher serait indéterminé.

Exemple 1.13

Le comportement du système d'information est indéterminé si une contrainte précise qu'un événement "opération à effectuer" déclenche exclusivement un processus "traitement 1" ou "traitement 2".

C. Une contrainte peut indiquer le nombre d'événements de classes différentes $E_1$, ..., $E_N$ qui doivent contribuer au déclenchement d'un même processus $P_1$.

Exemple 1.14

> Il faut 10 événements "arrivée d'une commande (du jour)" OU "sélection d'une commande (différée)" pour déclencher un seul processus "parcours en magasin".

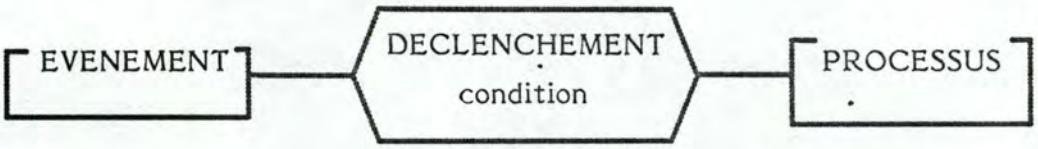
Comme il l'a été précisé plus haut, ces différentes expressions de comparaison peuvent intervenir dans les expressions booléennes plus complexes traduisant les modalités de déclenchement d'une même classe.

Exemple 1.15

> Il faut 10 événements "commande enregistrée" OU un événement "fin du jour" ET entre 1 et 10 événements "commande enregistrée" pour déclencher un seul processus "parcours en magasin".

**Spécification**

Une classe de **déclenchements**




est définie par

- le nom de la classe d'**événements** qui contribuent aux déclenchements ;
- le nom de la classe des **processus** qui sont déclenchés ;
- les **modalités (condition) de déclenchement** associées. Si aucune modalité n'est spécifiée, on suppose que
  - chaque événement contribue au déclenchement d'**un seul** processus
  - un processus est déclenché par la survenance d'**un seul** événement.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide d'une des deux formes syntaxiques suivantes :

a. DEF PROCESSUS nom-de-processus
DECLENCHE PAR nom-d'événement [QUAND nom-de-condition]

b. DEF EVENEMENT nom-d'événement
DECLENCHE nom-de-processus [QUAND nom-de-condition]

où une conditon de déclenchement est rédigée de la façon suivante

DEF CONDITION nom-de-condition
PREDICAT :
description du prédicat dans lequel les opérandes des expressions de comparaison peuvent être

1. nom-d'attribut DE nom-d'événement
2. DELAI ENTRE nom-d'événement ET nom-de-processus
3.a NOMBRE DE nom-d'événement DECLENCHANT LE MEME nom-de-processus
3.b NOMBRE DE nom-de-processus DECLENCHE PAR LE MEME nom-d'événement

Les classes de déclenchements de l'exemple I.6. peuvent donc être spécifiées de la façon suivante :

DEF PROCESSUS enregistrement-de-commande
DECLENCHE PAR arrivée-de-commande

DEF PROCESSUS archivage-de-commande
DECLENCHE PAR archivage-de-commande, fin-du-jour QUAND archivage-possible

DEF CONDITION archivage-possible
PREDICAT :
présence-signature DE arrivée-de-commande = VRAI ET DELAI ENTRE arrivée-de-commande ET archivage-de-commande >=
DELAI ENTRE fin-du-jour ET archivage-de-commande ET NOMBRE DE archivage-de-commande DECLENCHE PAR LE MEME fin-du-jour >= 0

La spécification des classes de déclenchements ayant trait à une même classe de processus constitue une **"pré-condition"** dynamique au déclenchement de ces processus puisqu'elle traduit l'état dans lequel le système d'information doit se trouver pour qu'un tel processus commence son exécution.

### I.2.6. RELATION D'ORDRE CHRONOLOGIQUE

Les associations de déclenchement et de survenance définissent une **relation d'ordre chronologique** partiel entre les événements et les processus. Une association de déclenchement exprime que la survenance d'un événement est suivie dans le temps du déclenchement d'un processus tandis qu'une association de survenance indique que le déclenchement d'un processus est suivi dans le temps de la survenance d'un événement.

Cette relation d'ordre chronologique est **non réflexive** puisqu'un événement ou un processus ne peut être son propre prédécesseur et, de la même façon, elle est **non symétrique** puisque si un événement ou un processus précède un autre, il ne peut donc lui succéder. Toutefois, cette dernière observation n'empêche nullement qu'un événement ou un processus précède un autre de la même classe traduisant ainsi l'existence d'un **circuit.** Par contre, cette relation est **transitive** puisqu'un événement ou un processus qui précède un autre, lui-même prédecesseurs d'un troisième, est également un prédécesseur du troisième.

Par récurrence, un événement ou un processus précède un autre lorsqu'il existe une succession d'événements et de processus qui, en fonction des déclenchements et des survenances, relient le premier et le second.
Enfin, cette relation est **partielle** car aucun ordre ne peut être déduit entre les événements dont la survenance est causée par le même processus ou entre les processus qui sont déclenchés par un même événement.

**Situations conventionnelles**

La spécification du comportement dynamique fonctionnel peut mettre en évidence les situations conventionnelles suivantes :

a. SEQUENCE lorsque chaque événement ou processus d'une classe précède systématiquement un événement ou un processus d'une autre classe ;

b. DECISION lorsque
   - une condition de déclenchement porte sur la valeur de certains attributs d'événements contribuant au déclenchement d'un processus,
   - une condition de survenance complète la spécification d'une classe de survenances.

c. PARALLELISME lorsque
   - un événement contribue au déclenchement de plusieurs processus d'une même classe ou de classes différentes,
   - un processus cause la survenance de plusieurs événements d'une même classe (cardinalité) ou de classes différentes ;

d. SYNCHRONISATION lorsque plusieurs éyénements d'une même classe (= ACCUMULATION) ou de classes différentes contribuent au déclenchement d'un même processus ;

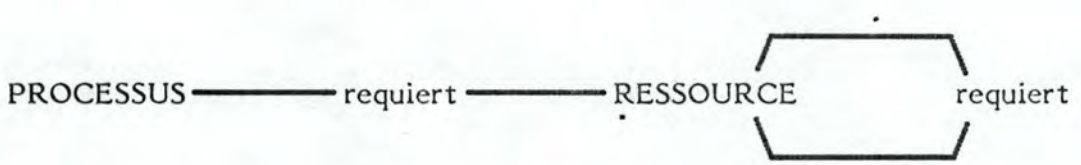
e. CIRCUIT lorsqu'un événement ou un processus précède un autre de la même classe. Ce circuit est **infini** lorsqu'aucun phénomène de décision n'intervient dans la succession des déclenchements et des survenances qui relient les deux éléments ;

f. INDETERMINISME lorsque différents processus sont candidats au déclenchement par la survenance d'un événement qui ne peut, selon ses modalités de déclenchement, contribuer qu'à un seul déclenchement.

## I.3. MODELISATION ET SPECIFICATION DES MOYENS REQUIS

Cette partie du modèle sert à caractériser les processeurs qui exécutent les processus. La spécification du comportement fonctionnel est, en effet et par définition, indépendant des moyens nécessaires à sa mise en oeuvre. Il importe, cependant, d'estimer si une solution conceptuelle est faisable, compte tenu des moyens humains, techniques, financiers et organisationnels dont dispose l'organisation. Dans la mesure où elles risquent d'avoir une influence sur le comportement du système d'information décrit, les **ressources requises** par les **processus** seront donc spécifiées (cf. préambule).

La figure suivante illustre les interactions entre les deux concepts de base du modèle proposé pour prendre en compte les moyens requis dans la spécification du comportement d'un système d'information :




Sachant que ce modèle est principalement destiné à décrire le comportement des systèmes d'information administratifs et de gestion, les ressources modélisées seront, surtout, celles de l'organisation dans laquelle s'insère le système d'information construit. Cette constatation et le fait qu'il s'agit d'un modèle couvrant l'étude d'opportunité et l'analyse fonctionnelle d'un projet justifient l'approche globale et agrégée qui a été adoptée. Cette approche est caractérisée par les hypothèses suivantes :

- un **processus** est considéré comme une "boîte noire" et **seule sa durée d'exécution estimée importe** ;
- une **ressource** est seulement caractérisée par une **capacité maximale disponible** et des **périodes de disponibilité** ;
- la **réquisition** d'une ressource par un processus est exprimée sous la forme de **fonction linéaire** de la durée d'exécution du processus et caractérisée par un taux de réquisition.

Après avoir proposé une vue synthétique des approches "ressource-processus" que l'on retrouve dans la littérature, cette section présentera le concept de ressource et ses interactions de réquisition avec les processus pour prendre en compte les moyens requis dans le comportement d'un système d'information. Cette section se terminera par une présentation récapitulative de l'évolution temporelle des processus compte tenu de la disponibilité des moyens qu'ils requièrent.

### I.3.1. Le concept de RESSOURCE dans la littérature

Dans le contexte de la modélisation des systèmes d'information, la prise en compte des moyens requis, c'est-à-dire l'utilisation de ressources physiques par les traitements, est généralement faite dans une optique d'évaluation - par simulation notamment - d'une solution détaillée et rares sont les modèles qui les intègrent dès la conception.
Les modèles qui suivent constituent l'exception, en particulier le modèle de [DUFOURD, 1980] qui aborde le problème dans le cadre de maquettes programmées de systèmes d'information.
Ainsi dans ce dernier modèle "une maquette de système d'information peut être définie comme un sytème complètement formalisé de processus coopérants par l'intermédiaire de structures de données partageables et concurrents pour l'obtention de ressources physiques. On distingue :

- les ressources actives qui sont celles qui modélisent les agents d'exécution de tout ou partie des processus ;
- les ressources passives qui représentent des moyens auxiliaires, consommables ou réutilisables, dont ont besoin les processus pour pouvoir se dérouler.

... les ressources d'un système d'information ne sont pas toujours disponibles pour lui : on fixe, alors, dans un calendrier les périodes pendant lesquelles une ressource peut être délivrée à un processus".

De façon assez semblable dans [BOYDSTON, 1983] et [SPEWAK, 1981], un modèle qui prend en compte la description des ressources physiques et leur utilisation par les traitements est proposé. Ils distinguent, également, les ressources actives - ou processeurs - et les ressources passives utilisées par les processeurs qui exécutent les processus. Par contre, la notion de calendrier de disponibilité n'est pas explicitement présente.

Nous retiendrons, principalement de ces modèles, et en particulier de celui de [DUFOURD, 1980]

- la prise en compte explicite, dès la conception d'un système d'information, des ressources physiques et de leur utilisation par les traitements comme une composante à part entière de la spécification du comportement d'un système d'information ;
- la notion de calendrier de disponibilité qui fixe les périodes de temps pendant lesquelles une ressource est disponible pour les traitements.

Par contre, notre modèle se distingue par une **prise en compte** plus **agrégée** des **ressources physiques** et de leur utilisation par les traitements en veillant à ne pas rentrer dans le détail fin des mécanismes d'allocation, de libération et d'ordonnancement.
Notre modèle se caractérise, en effet, par une approche dans laquelle l'élément central et intégrateur est le processus. Cela suppose que l'acquisition et la libération des ressources requises ne se font qu'au niveau d'un processus et qu'il n'a donc pas été prévu de pouvoir réserver une portion de ressource pour l'exécution "enchaînée" de plusieurs processus. Toute ressource requise par un processus est considérée comme devant être acquise en début d'exécution et libérée au terme de celle-ci.
Cette approche a été choisie pour conserver notre hypothèse qu'un processus est, à ce niveau de spécification,

- une entité **indépendante** (des autres processus) ;
- une "boîte noire" dont on ne précise que le comportement externe - via les événements - en ignorant le contenu de sa procédure.

L'avantage principal procuré par cette modélisation fort agrégée réside dans le fait que ce modèle peut être utilisé très tôt dans le cycle de vie d'un projet (étude d'opportunité) et permettre une première évaluation (cf. chapitre 2 et 3) du comportement spécifié.

## I.3.2. RESSOURCE

**Modélisation**

Une ressource représente un certain volume de moyens physiques identiques dont

- l'organisation doit disposer,
- une partie peut être requise par un traitement pour pouvoir s'exécuter,
- l'indisponibilité totale ou partielle peut entraîner le blocage temporaire d'un traitement qui la requiert.

Seules sont, en effet, prises en considération les ressources qui risquent d'avoir une influence sur le comportement du système d'information décrit soit parce que leur volume disponible est limité, soit parce qu'elles ne sont disponibles qu'à certains moments.

Les moyens physiques représentés par une ressource peuvent soit être **utilisés** par certains traitements et remis à la disposition du système après leur utilisation, soit être **consommés** par les traitements et non restitués par ceux-ci au terme de leur exécution.

Toute ressource est définie par un **nom** identifiant qui correspond à une unité de ressource réutilisable dans le premier cas, à une unité de ressource consommable dans le second.

La **capacité disponible** d'une ressource est une propriété qui indique la quantité de ressource effectivement disponible pour les traitements à un moment donné.

Le **calendrier de disponibilité** d'une ressource est une propriété qui fixe, pour chaque **période** pendant laquelle la ressource est disponible, la **capacité** ou volume maximal que peuvent se partager les processus qui la requièrent. Une période de disponibilité est exprimée sous la forme d'intervalles de temps qui se reproduisent périodiquement dans une période plus large. Ces intervalles de temps sont exprimés en unités temporelles qui font référence à une décomposition convenue du temps. Le calendrier de disponibilité constitue une **contrainte** (de disponibilité) qui porte sur la capacité disponible en fixant la valeur maximum qu'elle peut prendre dans la période considérée.

Exemple 1.16

> Un service dont le nombre d'employés et les horaires de travail risquent d'être critiques peut être représenté par une ressource (réutilisable) :

"employé service" dont la capacité est de
4 (employés service) disponibles de 9 à 12 heures le lundi,
6 (employés service) disponibles de 9 à 16 heures du mardi au vendredi.

Un ordinateur dont les performances sont fort dégradées au-delà d'un nombre connu d'utilisateurs simultanés peut-être représenté par une ressource (réutilisable) :

"point d'entrée CPU" dont la capacité est de
10 (points d'entrée CPU) disponibles 24 heures sur 24.

Une mémoire centrale dont la taille réduite risque d'être critique peut être représentée par une ressource (réutilisable) :

"K-octets mémoire" dont la capacité est de
1024 (K-octets mémoire) disponibles 24 heures sur 24.

Un réseau dont le taux de transfert est exprimé en nombre de bits transférés par seconde peut être représenté par une ressource (consommable) :

"M-bits transférés par seconde" dont la capacité est de
1.2 (M-bits transférés par seconde) disponibles 24 heures sur 24.

**Spécification**

Une **ressource**

| RESSOURCE |
|---|
| disponiblité |

est définie par

- un **nom** ;
- un calendrier de **disponibilité** qui précise la capacité disponible par période de temps. Si aucune période n'est spécifiée, on suppose que la capacité spécifiée est toujours disponible pour les processus requérants (cf. I.3.3.).

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide des instructions suivantes

```
DEF RESSOURCE                          nom-de-ressource
    CAPACITE valeur [DISPONIBLE PENDANT période]

où une période de disponibilité est spécifiée de la façon suivante :

DEF PERIODE                            nom-de-période
    DE constante-horaire A constante-horaire DANS nom-de-période
```

La ressource "employé service" de l'exemple 1.16 peut donc être spécifiée comme suit :

```
DEF RESSOURCE employé-service
   CAPACITE 4 DISPONIBLE PENDANT lundi-matin
            6 DISPONIBLE PENDANT mardi-au-vendredi
   DEF PERIODE lundi-matin
       DE 9H A 12H DANS lundi
   DEF PERIODE lundi
       DE 0H A 24H DANS semaine
   (...)
```

### I.3.3. REQUETE

**Présentation générale**

Qu'elles soient réutilisables ou consommables, les ressources sont considérées de la même manière par les traitements : une certaine quantité de ressource peut être requise, acquise et libérée par un processus pour pouvoir s'exécuter.

Une **requête** représente l'utilisation d'une **partie** de ressource - par un processus - pendant une certaine **durée.** Une requête est caractérisée par des propriétés qui précisent la quantité de ressource requise et situent l'évolution de la requête dans le temps.

La **durée de réquisition** indique la période pendant laquelle la portion de ressource est requise. Elle correspond, normalement, à la **durée d'exécution** du processus requérant. Cette durée d'exécution est une estimation du temps requis par le processus pour exécuter les règles de sa procédure.

Un processus peut, cependant, être ou non **interruptible.** Lorsqu'un processus est **non interruptible,** cela signifie que la durée de réquisition de ses requêtes correspond exactement à sa durée d'exécution car, une fois acquises, les ressources ne peuvent être libérées qu'au terme normal de son exécution. Par contre, un processus **interruptible** signifie que la durée de réquisition de ses requêtes peut être inférieure à sa durée d'exécution car les ressources peuvent éventuellement être libérées avant la fin normale de son exécution. Une requête et donc un processus peut être interrompu

- soit parce que la ressource est requise par un autre processus plus prioritaire (cf. règle de priorité I.3.4), par un mécanisme de préemption,
- soit parce que la ressource devient indisponible pour toute utilisation en fonction de son calendrier de disponibilité. Cette indisponibilité résulte alors d'une diminution de sa capacité maximale disponible - qui devient même, éventuellement, nulle - pour une période donnée.

Dans le cas d'une interruption, puisque le processus n'est pas terminé et reste concurrent pour l'obtention des ressources qu'il utilisait, tout se passe comme si de nouvelles requêtes étaient initialisées avec les mêmes propriétés que les requêtes initiales, mais avec une durée de réquisition ajustée pour tenir compte du temps pendant lequel le processus a déjà été en activité.

Le **taux de réquisition** détermine la quantité de ressource requise par le processus. Ce taux de réquisition a pour base la durée d'exécution du processus requérant. Cela signifie que la ressource est requise par le processus pendant toute la durée d'exécution du processus.

La spécification de la durée d'exécution d'un processus et du taux de réquisition de ses requêtes revient à considérer qu'à condition de pouvoir disposer de quantités de ressources équivalentes aux taux de réquisition, l'exécution du processus correspond à la durée d'exécution spécifiée du processus.

Les **dates de réquisition**, d'**acquisition** et de **libération** précisent respectivement les instants où la requête a été créée, la portion de ressource a été acquise par le processus, la portion de ressource acquise a été libérée par le processus. Entre la date de réquisition et celle d'acquisition, la requête et donc le processus sont en attente, tandis que pendant la durée de réquisition, entre la date d'acquisition et celle de libération, la quantité de ressource requise est à la disposition du processus qui est en activité.

On distingue, toutefois, 4 façons de représenter une requête selon que la réquisition est faite directement ou non par le processus et dans les deux cas selon qu'une même portion de ressource requise est partagée ou non par plusieurs processus.

### I.3.3.1. Réquisition directe

**Modélisation**

Une association de réquisition directe représente la réquisition, l'acquisition et la libération d'une portion de ressource directement par un processus. Cela signifie qu'un processus requiert pour son seul usage la quantité de ressource correspondant au taux de réquisition.

Toute assocation de réquisition directe appartient à la classe des réquisitions directes qui relient les processus d'une même classe à une même ressource et sont caractérisées par un même taux de réquisition.

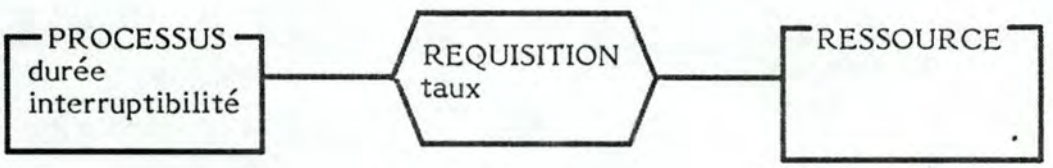
Exemple 1.17

Un processus d'"enregistrement de commande" dure en moyenne 10 minutes à condition de pouvoir disposer (pendant ces 10 minutes)

d'1 "employé service",
d'1 "point d'entrée CPU",
de 64 "K-octets mémoire",
et de 0.01 "M-bits transférés par seconde". (représentant le transfert de 6 millions de bits : 6 millions en 10 minutes correspondent à 10.000 bits en une seconde).

**Spécification**

Une classe de **réquisitions directes**




est définie par

- le nom de la classe des **processus** requérants ;
- le nom de la **ressource** requise ;
- le **taux** de réquisition. Si ce taux n'est pas spécifié, on suppose que chaque processus requiert **une** unité de la ressource. Le taux peut être exprimé sous la forme d'un nombre ou d'une distribution de probabilité.

Cette spécification suppose cependant que la **durée d'exécution** et le caratère **interruptible** ou non des processus requérants aient été spécifiés. En l'absence d'une spécification contraire, on suppose que les processus sont NON interruptibles. La durée d'exécution peut également être exprimée sous la forme d'un nombre ou d'une distribution de probabilité.

Dans le contexte de l'environnement logiciel adopté, les spécifications qui caractérisent une classe de processus sont rédigées à l'aide des instructions suivantes :

```
DEF PROCESSUS        nom-de-processus
    ACTIF PENDANT    durée
    INTERRUPTION     {AUTORISEE | INTERDITE}
```

Une classe de réquisitions directes peut être spécifiée à l'aide d'une des deux formes syntaxiques équivalentes suivantes :

| | | |
|---|---|---|
| a. DEF | PROCESSUS | nom-de-processus |
| | REQUIERT | taux nom-de-ressource |
| b. DEF | RESSOURCE | nom-de-ressource |
| | REQUISE PAR | nom-de-processus AU TAUX DE taux |

Les instructions suivantes spécifient les classes de réquisitions de l'exemple 1.17 :

```
DEF PROCESSUS        enregistrement-de-commande
    ACTIF PENDANT    10 min.
    REQUIERT         1 employé-service,
                     1 point-d'entrée-CPU,
                     64 K-octets-mémoire,
                     0.01 M-bits-transférés-par-secondes
```

### I.3.3.2. Réquisition directe partagée

**Modélisation**

Une association de réquisition directe partagée représente la requête d'une portion **commune** de ressource utilisée par **tous** les processus - en cours d'exécution - d'une même classe.
Contrairement à une réquisition directe où chaque processus en cours d'exécution doit disposer d'une portion différente de la ressource, dans le cas d'une réquisition partagée, **tous** les processus d'une même classe en cours d'exécution au **même moment** partagent la **même** portion de la ressource (requête non additive).
Toute réquisition ne donne donc plus systématiquement lieu à une acquisition et à une libération puisque l'acquisition ou la libération effective de la portion de ressource requise est nécessaire, uniquement, s'il n'existe aucun autre processus de la même classe en cours d'exécution.

Toute association de réquisition directe partagée appartient à la classe des réquisitions directes partagées qui relient les processus d'une même classe à une même ressource et sont caractérisées par un même taux de réquisition.

Exemple 1.18

En plus des 64 "K-octets" que requiert chaque "enregistrement de commande" tous les "enregistrements de commande" qui

s'exécutent simultanément se partagent (une zone commune de) 128 "K-octets mémoire" (correspondant, par exemple, à la partie de mémoire occupée par un code réentrant).

**Spécification**

Les spécifications d'une classe de réquisitions directes partagées sont rédigées de manière identique à celles d'une classe de réquisitions directes (cf. I.3.3.1.). Seuls les mots REQUIERT et REQUIS PAR ont respectivement été substitués par PARTAGE et PARTAGE PAR. Toutefois, le taux de réquisition doit toujours être exprimé sous la forme d'un nombre.

La classe de réquisitions directes partagées de l'exemple 1.18 peut donc être spécifiée comme suit :

| | |
|---|---|
| DEF PROCESSUS | enregistrement-de-commande |
| PARTAGE | 128 K-octets-mémoire |

### I.3.3.3. Réquisition auxiliaire

**Modélisation**

Une association de réquisition auxiliaire représente la requête d'une portion de ressource - dite auxiliaire - à l'occasion d'une requête portant sur une ressource - dite principale, c'est-à-dire lorsque celle-ci est elle-même requise (directement ou non) par un processus.
Les valeurs des propriétés d'une requête auxiliaire sont, principalement, dérivées de celles prises par les propriétés de la requête principale.
La **date de réquisition** et la **durée de réquisition** correspondent à celles de la requête principale.
Par contre, le **taux de réquisition** fixe la quantité de ressource auxiliaire qu'UNE unité de ressource principale requiert. En d'autres termes, la quantité de ressource auxiliaire requise est **proportionnelle** à la quantité de ressource principale requise.

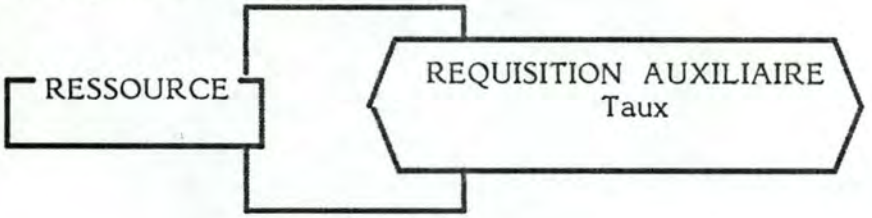
Toute association de réquisition auxiliaire appartient à la classe des réquisitions auxiliaires qui relient une ressource à une autre et sont caractérisées par un même taux de réquisition.

Exemple 1.19

Chaque "employé service", lorqu'il est requis par un traitement, doit disposer d'un "terminal de pool" (tant que l'employé est occupé).

**Spécification**

Une classe de **réquisitions auxiliaires**




est définie par

- le nom d'une **ressource** - dite principale ;
- le nom d'une autre **ressource** - dite auxiliaire ;
- un **taux** de réquisition. Si ce taux n'est pas spécifié, on suppose qu'**une** unité de ressource principale requiert **une** unité de ressource auxiliaire. Ce taux peut être exprimé sous la forme d'un nombre ou d'une distribution de probabilité.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide d'une des deux formes syntaxiques - équivalentes - suivantes :

| | | |
|---|---|---|
| a. DEF RESSOURCE | | nom-de-ressource |
| | REQUIERT | taux nom-de-ressource |
| b. DEF RESSOURCE | | nom-de-ressource |
| | REQUISE PAR | nom-de-ressource AU TAUX DE taux |

La classe de réquisitions auxiliaires de l'exemple 1.19 peut être spécifiée comme suit :

```
DEF RESSOURCE        employé-service
    REQUIERT         1  terminal-du-pool
```

### I.3.3.4. Réquisition auxiliaire partagée

**Modélisation**

Une association de réquisition auxiliaire partagée représente la requête d'une portion de ressource - dite auxiliaire - dès qu'une ressource - dite principale - fait l'objet d'une ou plusieurs requêtes.
Contrairement à une réquisition auxiliaire, dans le cas d'une réquisition auxiliaire partagée, la quantité requise de la ressource auxiliaire est **non proportionnelle** à la quantité requise de la ressource principale.
Mais de la même façon qu'une réquisition directe partagée, une réquisition auxiliaire partagée ne donne plus systématiquement lieu à une acquisition et une libération. L'acquisition ou la libération de la portion de ressource requise a lieu uniquement si la ressource principale n'est pas déjà requise par ailleurs.

Toute association de réquisition auxiliaire partagée appartient à la classe des réquisitions auxiliaires partagées qui relient une ressource à une autre et sont caractérisées par un même taux de réquisition.

Exemple 1.20

Dès qu'un "employé service" est requis, le service requiert (l'attention d') 1 "superviseur service" (indépendamment du nombre d'employés occupés).

Si un fichier F1 peut être accédé simultanément en consultation par plusieurs processus PC1, PC2, ... ou PCM mais devient non partageable dès qu'un seul processus PM1, PM2 ... ou PMM y accède pour le modifier, on représentera cette situation de la façon suivante :

La ressource "fichier F1",
dont la capacité est 1
requiert 1000 "pages disque" par exemple,
est requise par PM1, PM2 ... ou PMM
et est partagée par la ressource "consultations fichier F1" et

la ressource "consultations fichier F1"
est requise par PC1, PC2 ... PCM et
partage donc le "fichier F1".

**Spécification**

Les spécifications d'une classe de réquisitions auxiliaires partagées sont rédigées de manière identique à celles d'une classe de réquisitions auxiliaires (cf. I.3.3.3.). Seuls les mots REQUIERT et REQUIS PAR ont respectivement été substitués par PARTAGE et PARTAGE PAR. Toutefois, le taux de réquisition doit toujours être exprimé sous la forme d'une constante.

Le partage du "superviseur service" de l'exemple 1.20 peut ainsi être spécifié comme suit :

```
DEF RESSOURCE      employé-service
    PARTAGE        1   superviseur-service
```

## I.3.4. REGLES DE PRIORITE

**Modélisation**

Lorsqu'une ressource est temporairement indisponible et ne peut servir simultanément toutes les requêtes, des règles d'ordonnancement (ou de gestion de file d'attente) permettent de choisir parmi les requêtes celles qui doivent être servies avant d'autres.
Ces règles traduisent des phénomènes de priorité entre les requêtes visant soit

à **privilégier** systématiquement certaines requêtes par rapport à d'autres **pour des raisons fonctionnelles,** soit à assurer une **optimisation** dans l'utilisation des ressources.

Dans la mesure où ce modèle a pour objectif la **représentation au stade de la conception** (cf. préambule) du comportement d'un système d'information, il **n'intègre pas** la représentation des mécanismes d'**ordonnancement** et de gestion des files d'attente ayant pour but d'optimiser l'utilisation des ressources. Ces règles et stratégies d'ordonnancement seront spécifiées et mises en oeuvre dans les étapes de réalisation du système d'information.
Par contre, certaines règles de **priorité fonctionnelle** peuvent être mentionnées en signalant que les processus d'une classe P1 sont **plus prioritaires** que ceux d'une classe P2. Ces règles constituent, alors, des contraintes à respecter lors du choix des règles d'ordonnancement et de gestion des files d'attente.

Ces règles de priorité peuvent cependant avoir des **effets différents** selon que les **processus moins prioritaires** sont **interruptibles** ou non (cf. I.3.3.1.). En effet, si le processus moins prioritaire

a. n'est pas interruptible :
   la règle ne s'applique alors qu'aux processus en attente. Dans ce cas, un processus - même moins prioritaire - qui est en activité ne **peut pas être interrompu** par un processus plus prioritaire. En d'autres termes, un processus qui démarre - parce que les ressources requises étaient disponibles - se termine après sa durée estimée d'exécution ;

b. est interruptible (mécanisme de préemption) :
   la règle s'applique alors aux processus en attente **et en activité.** Cela signifie qu'un processus en activité peut alors être **interrompu par un processus plus prioritaire** qui requiert - avec **préemption** - une ressource utilisée par le processus moins prioritaire.

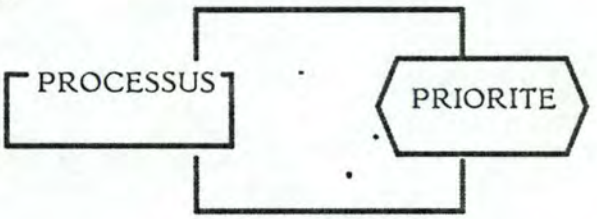
Exemple 1.21

> Les processus d'"enregistrement de commandes différées" sont plus prioritaires que les processus d'"enregistrement des commandes du jour".
> Lorsqu'un processus d'"enregistrement d'une commande différée" est déclenché et requiert une ressource non disponible car utilisée par un ou plusieurs processus d'"enregistrement de commandes du jour" :
> a. si les processus d'"enregistrement de commandes du jour" sont **NON interruptibles,** alors cette règle de priorité ne s'applique qu'aux processus "enregistrement de commandes du jour" en attente ;

b. par contre, s'ils sont **interruptibles,** alors le processus "enregistrement d'une commande différée" pourra interrompre l'exécution d'un "enregistrement d'une commande du jour" pour pouvoir disposer de ses ressources.

**Spécification**

Une **règle de priorité**




est définie par

- le nom d'une classe de processus PLUS prioritaires ;
- le nom d'une classe de processus MOINS prioritaires.

Dans le langage associé, cette spécification peut être rédigée à l'aide d'une des deux formes syntaxiques - équivalentes - suivantes :

```
a. DEF PROCESSUS          nom-de-processus
   MOINS PRIORITAIRE QUE  nom-de-processus

b. DEF PROCESSUS          nom-de-processus
   PLUS PRIORITAIRE QUE   nom-de-processus
```

La règle de l'exemple 1.21 peut donc être rédigée de la manière suivante :

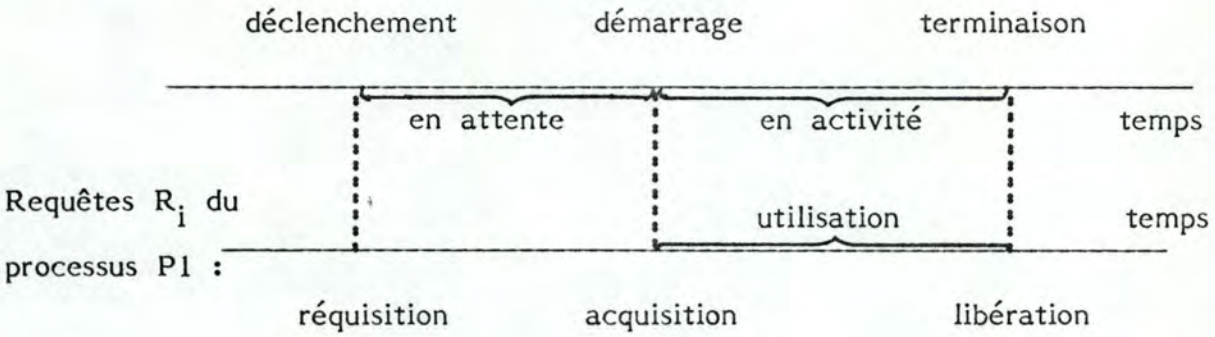
```
DEF PROCESSUS    enregistrement-de-commandes-du-jour
  MOINS PRIORITAIRE QUE  enregistrement-de-commandes-
                         différées
```

## I.3.5. EVOLUTION TEMPORELLE DES PROCESSUS ET DES REQUETES

Le modèle adopté suppose un comportement convenu des processus en fonction des ressources qu'ils requièrent.

Le schéma suivant illustre

- le comportement convenu d'un processus qui peut disposer de la totalité des ressources requises pendant toute sa durée d'exécution ;
- l'évolution parallèle des requêtes associées à ce processus.




Déclenchement et réquisition

Un nouveau processus est déclenché (= créé) par la survenance d'un ou plusieurs événements. La **date de déclenchement** correspond à ce moment. Au même moment, les requêtes du processus sont créées (= réquisition) et la **date de réquisition** est donc identique à la date de déclenchement du processus requérant.

Démarrage et acquisition

Le démarrage du processus, c'est-à-dire le début de son exécution, a lieu lorsque le processus peut disposer des ressources qu'il requiert. La **date de démarrage** est une propriété temporelle du processus qui localise dans le temps ce démarrage. Elle correspond également à la **date d'acquisition** de chaque requête associée au processus.
Entre la date de déclenchement et celle de démarrage, le **processus est en attente** - pour cause d'indisponibilité temporaire de certaines ressources requises. Cette attente est identique à celle des requêtes du processus entre leurs dates de réquisition et d'acquisition.

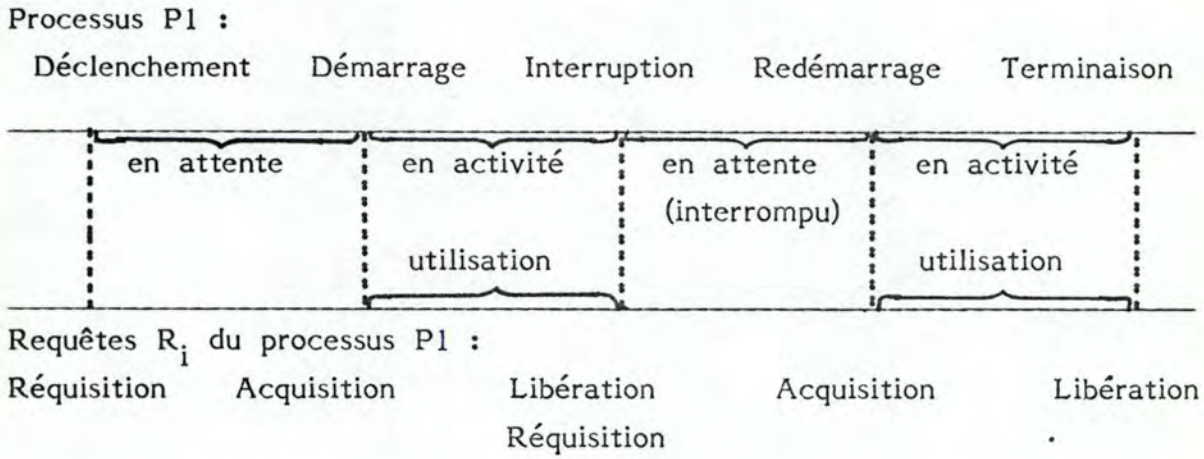
Terminaison et libération

Après une durée - estimée - d'exécution, qui correspond au temps requis pour exécuter les règles de sa procédure, le processus se termine. La **date de terminaison** correspond à cette fin d'exécution et est identique à la **date de libération** des requêtes associées au processus.
Entre la date de démarrage et celle de terminaison, le **processus est en activité** et de façon analogue les dates d'acquisition et de libération des requêtes du processus déterminent la période pendant laquelle les portions de ressources requises sont utilisées et donc indisponibles pour d'autres processus.

Après la date de terminaison, le processus a libéré toutes les ressources qu'il avait requises et n'a plus aucun effet immédiat sur le système si ce n'est indirectement via les événements dont il a causé la survenance en cours d'exécution.

Le schéma suivant illustre par contre

- le comportement convenu d'un processus qui a été **interrompu** parce que certaines des ressources qu'il utilisait ont été requises - avec préemption - par un autre processus plus prioritaire ou sont devenues indisponibles en fonction de leur calendrier de disponibilité ;
- l'évolution parallèle des requêtes associées à ce processus interrompu.




Interruption et libération/réquisition

Un processus est interrompu parce que certaines des ressources qu'il utilisait ont été requises - avec préemption - par un processus plus prioritaire. Il peut également être interrompu parce que certaines des ressources qu'il utilisait sont devenues indisponibles pour toute utilisation en fonction de leur calendrier. Une **date d'interruption** est une propriété temporelle qui localise dans le temps cette interruption. Contrairement aux dates de déclenchement, de démarrage et de terminaison qui sont uniques, plusieurs dates d'interruption peuvent être associées à un même processus et traduisent ainsi le comportement d'un processus qui a dû être interrompu plus d'une fois.

Lors d'une interruption, les requêtes du processus sont libérées et leur **date de libération** correspond alors à la date d'interruption du processus. De plus, comme le processus reste concurrent pour l'acquisition des mêmes ressources, de nouvelles requêtes sont créées (= réquisition) avec des **dates**

**de réquisition** elles aussi identiques à la date d'interruption du processus.
Entre la date de (re-) démarrage et celle d'interruption, le processus est en activité et de la même façon, les requêtes du processus sont en activité (ou en service) entre leurs dates d'acquisition et de libération.

Redémarrage et acquisition

Dès que le processus peut à nouveau disposer des ressources qu'il requérait, il redémarre et une **date de redémarrage** repère cet instant. Plusieurs dates de redémarrage peuvent être associées à un même processus puisque normalement à chaque interruption correspondra un redémarrage.
Lors d'un redémarrage, le comportement d'un processus et de ses requêtes est totalement identique à celui décrit lors d'un démarrage normal si ce n'est que la durée estimée d'exécution a dû être ajustée pour tenir compte du temps pendant lequel le processus a déjà été en activité.

## I.4. MODELISATION ET SPECIFICATION DE LA CHARGE ESTIMEE

Le dernier aspect qui influence le comportement d'un sytème d'information est la charge estimée du système, c'est-à-dire le nombre d'événements et de processus prévus à un moment donné.
L'estimation de la charge d'un système d'information est faite en précisant un échéancier de survenances pour toutes les classes d'événements externes (cf. I.2.3.) au sytème d'information.

### I.4.1. ECHEANCIER

**Modélisation**

Les événements externes portent à la connaissance du système d'information des changements d'état qui sont provoqués par l'environnement de ce dernier.
L'**échéancier** d'une classe d'événements externes est une propriété qui fixe le nombre d'événements et leur fréquence de survenance par périodes de temps.

Exemple 1.22

- Entre 200 et 300 "arrivées de commande" surviennent à 9 h et 14 h tous les jours pour représenter l'arrivée de ces commandes aux courriers du matin et de l'après-midi.

- Entre 1 et 10 "commandes différées sélectionnées" arrivent en moyenne toutes les 20 minutes (selon une loi de poisson, par exemple) pendant les heures de travail, c'est-à-dire entre 9 et 12 h et entre 14 et 17 h tous les jours ouvrables.

**Spécification**

Un **échéancier**

| EVENEMENT |
|---|
| échéancier |

est défini par un ensemble de

- volumes et fréquences par périodes de temps.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide des instructions suivantes :

```
DEF EVENEMENT      nom-d'événement
    SURVIENT       nombre FOIS [TOUTES LES fréquences]
                   PENDANT période
```

Les deux échéanciers de l'exemple 1.22 sont donc spécifiés de la façon suivante :

```
DEF EVENEMENT      arrivée-de-commande
    SURVIENT       entre-200-et-300 FOIS PENDANT courrier
DEF PERIODE        courrier
    A 9H, 14H      DANS semaine-de-travail
DEF EVENEMENT      commande-différée-sélectionnée
    SURVIENT       entre-1-et-10 FOIS TOUTES LES en-moy-
                   enne-20-min. PENDANT heures-de-travail
DEF PERIODE        heures-de-travail
    DE 9H A 12H,14H A 17H DANS semaine-de-travail
DEF PARAMETRE      en-moyenne-20-min.
    DISTRIBUTION NEGEXP AVEC PARAMETRE 20 min
```

## I.5. MODELISATION ET SPECIFICATION DES PERFORMANCES ATTENDUES

Cet aspect de la spécification du comportement d'un système d'information correspond à l'établissement des performances attendues de ce système compte tenu de la spécification du comportement fonctionnel, de la prise en compte des moyens requis et de l'estimation de la charge.

### I.5.1. DELAI entre événements successeurs

**Modélisation**

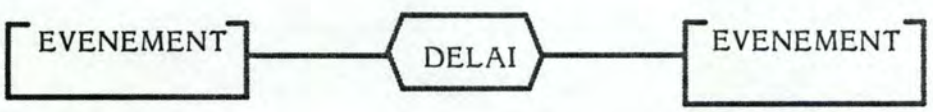
La spécification des performances attendues du comportement d'un système porte essentiellement sur le **délai** compris entre la survenance des événements d'une classe E1 et celle des événements d'une classe E2 qui leur succèdent dans le temps (cf. I.2.6.)

Exemple 1.23

> Le délai entre la survenance d'une "commande du jour" et la survenance d'une "commande expédiée" doit être inférieur à 24 h dans 80 % des cas.
>
> Le délai entre la survenance d'une "réquisition à préparer" et la survenance d'une "commande préparée" doit être compris entre 4 h et 12h.

**Spécification**

Un **délai inter-événements**




est défini par

- le nom d'une classe d'**événements** ;
- le nom d'une seconde classe d'**événements** (qui succèdent à ceux de la première classe) ;
- la valeur du **délai** compris entre la survenance d'un événement de la première classe et un autre de la seconde classe - qui succède au premier.

Dans le contexte de l'environnement logiciel adopté, cette spécification est rédigée à l'aide d'une des deux formes syntaxiques suivantes :

| | | | |
|---|---|---|---|
| a. DEF EVENEMENT | nom-d'événement | | |
| SUCCEDE A | nom-d'événement | APRES | délai |
| b. DEF EVENEMENT | nom-d'événement | | |
| PRECEDE | nom-d'événement | DE | délai |

Les performances attendues exprimées dans l'exemple 1.23 peuvent donc être spécifiées comme suit :

DEF EVENEMENT commande-du-jour
PRECEDE commande-expédiée DE environ-24H-dans-80-%

DEF EVENEMENT commande-préparée
SUCCEDE A réquisition à préparer APRES entre-4-et-12H

## I.6. MODELISATION ET SPECIFICATION DE LA DECOMPOSITION EN SOUS-SCHEMAS

Un **schéma de comportement** est une description du comportement d'un système d'information ou d'une partie de celui-ci faite à l'aide des concepts exposés dans les sections précédentes. Un tel schéma comprend donc les 4 aspects qui caractérisent le comportement :

- comportement fonctionnel proprement dit ;
- moyens requis ;
- charge estimée ;
- performances attendues.

Dans un tel schéma, une classe d'**événements initiaux** est une classe qui regroupe des événements externes, c'est-à-dire dont la survenance n'est pas causée par un processus d'une classe du schéma. Une classe d'**événements terminaux** est une classe qui regroupe des événements qui ne contribuent à aucun déclenchement de processus d'une classe du schéma.

L'établissement des spécifications du comportement procédant souvent par affinements successifs, la possibilité de désagréger progressivement un schéma en **sous-schémas** - et de contrôler cette désagrégation (cf. 4.3.3.) - est d'une grande importance dans un contexte méthodologique. Cette décomposition - contrôlée - en sous-schémas peut en quelque sorte être assimilée à celle prévue dans les approches "Flux de données" (DATA FLOW) représentées principalement par [DE MARCO, 1978].

## I.6.1. SOUS-SCHEMA

**Modélisation**

Un **sous-schéma** S1 d'un schéma S2 est un schéma qui décrit le comportement des processus d'une classe du schéma S2. Toute procédure, puisqu'elle décrit le comportement d'une partie d'un système d'information, peut en effet être décrite à un niveau plus désagrégé par un (sous-)schéma de comportement. De la même façon, tout schéma peut à son tour être représenté par une classe de processus d'un schéma plus agrégé.

De plus, lorsqu'un schéma S1 représente le comportement d'une classe de processus P1 d'un schéma S2, la définition de la classe des processus P1 et celles des classes de survenances et de déclenchements dans lesquelles interviennent les processus P1 constituent les spécifications du sous-schéma S1, c'est-à-dire des contraintes à respecter lorsqu'on définit - plus finement - le comportement des processus P1.

Exemple I.24.

Si, dans un schéma S1, une classe de processus est spécifiée de la façon suivante :

```
DEF PROCESSUS          préparation-réquisition
    DECLENCHE PAR      réquisition-à-préparer
    CAUSE              1 commande-préparée
```

Le schéma suivant décrit le comportement dynamique fonctionnel des processus de cette classe et constitue donc un sous-schéma du schéma initial S1 :

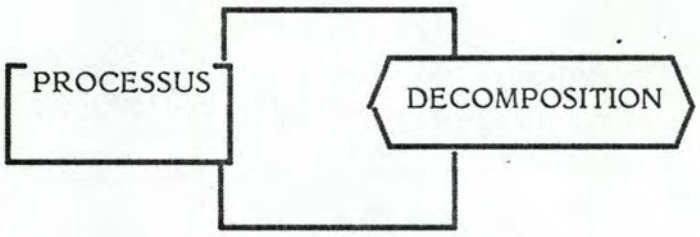
```
DEF PROCESSUS          ordonnancement-série
    DECLENCHE PAR      réquisition-à-préparer
                       QUAND série-constituée
    CAUSE              1 série-à-préparer

DEF CONDITION          série-constituée
    PREDICAT :
      NOMBRE DE réquisition-à-préparer DECLENCHANT
      LE MEME    ordonnancement-série = 10

DEF PROCESSUS          préparation-série
    DECLENCHE PAR      série-à-préparer
    CAUSE              10 commande-préparée
```

On exposera de manière plus complète au chapitre 4 comment vérifier la cohérence de cette décomposition. On peut cependant déjà remarquer que :

- les événements initiaux (réquisition-à-préparer) du sous-schéma sont ceux qui déclenchent les processus (préparation-réquisition) ;
- les événements terminaux (commande-préparée) du sous-schéma sont ceux dont la survenance est causée par les mêmes processus.

**Spécification**

La **décomposition** de processus en processus plus désagrégés




est définie par

- le nom d'une classe de **processus** ;
- le nom d'une autre classe de **processus,** qui sont des sous-processus des premiers.

Dans le contexte de l'environnement logiciel adopté, cette spécification peut être rédigée à l'aide d'une des deux formes syntaxiques - équivalentes - suivantes :

| | | |
|---|---|---|
| a. DEF | PROCESSUS | nom-de-processus |
| | SOUS-PROCESSUS | nom-de-processus |
| b. DEF | PROCESSUS | nom-de-processus |
| | SOUS-PROCESSUS DE | nom-de-processus |

La spécification des deux classes de processus de l'exemple 1.24 sera donc complétée de la façon suivante :

```
DEF PROCESSUS                  préparation-réquisition
    SOUS PROCESSUS SONT        ordonnancement-série,
                               préparation-série
```

- CHAPITRE II -

# OUTIL LOGICIEL D'EVALUATION DU COMPORTEMENT DES SYSTEMES D'INFORMATION

## II.1. INTRODUCTION

Ce chapitre décrit les fonctions et l'architecture d'un outil logiciel dédicacé au modèle proposé dans le chapitre 1. L'objectif de cet outil est d'**étudier le comportement** d'un système d'information en fonctionnement. Sa principale application dans la contruction d'un schéma sera la **vérification de la faisabilité** du schéma telle que définie au chapitre suivant.

Deux idées prévalent à la conception et à la réalisation de cet outil [BODART, 1977] :

1. Une technique de **simulation informatique** constitue un moyen privilégié pour étudier le comportement d'un système d'information et en évaluer a priori les performances ;
2. L'intégration de cet outil dans l'environnement logiciel de spécification adopté (cf. préambule) est assurée par la **génération automatique** d'un programme de simulation au départ des seules spécifications enregistrées dans la base de données. Par rapport aux approches classiques de simulation séparée ou autonome, cette technique de génération automatique permet de
   - supprimer les risques de divergence entre les spécifications fonctionnelles et le programme de simulation équivalent ;
   - réduire le temps de réponse aux changements de spécifications.

Ces 2 idées avaient déjà été partiellement évoquées dans [KONSYNSKI, 1976].

Cette approche par simulation au niveau des spécifications fonctionelles se distingue également des approches plus conventionnelles [BEYERLE, 1977] [DAHLE, 1973] [KREUTZER, 1976] [SANGUINETTI, 1979] [WILLIS, 1978] dans lesquelles une simulation n'est souvent envisagée qu'à un stade avancé de l'analyse de conception détaillée. L'introduction de technique de simulation dès les étapes initiales de conception (étude d'opportunité et analyse fonctionnelle) permet d'envisager très tôt différentes solutions alternatives dans les spécifications du comportement d'un système d'information et d'en étudier a priori les conséquences. Cet outil autorise donc et favorise même une approche expérimentale dans la construction d'un schéma de comportement. Enfin, davantage encore qu'un outil pour l'informaticien, il s'agit d'un moyen privilégié aux mains des organisateurs pour évaluer les conséquences sur l'organisation résultant de la mise en place du système d'information projeté.

Cet outil présente certaines analogies avec trois outils développés par ailleurs :

1. MAESTRO [DUFOURD, 1980] ;
2. SARA [RAZOUK, 1979] [WINCHESTER, 1980] ;
3. SDL/SDA [SPEWAK, 1981] [BOYDSTON, 1983].

1. [DUFOURD, 1980] propose le **système MAESTRO** qui permet une mise au point rapide de maquettes logicielles pour évaluer les systèmes d'information des organisations. Une telle maquette est vue comme "un ensemble de processus coopérant par l'intermédiaire de structures de données communes partageables et concurrents pour l'obtention de ressources physiques". MAESTRO est en fait une collection de classes et de procédures SIMULA67 [DAHLE, 1970] qui ont été ajoutées à l'environnement de simulation de SIMULA67 pour faciliter l'écriture de maquettes de systèmes d'information. Une maquette rédigée à l'aide de ces primitives de haut niveau peut notamment aider à analyser le comportement d'un système, prévoir ses performances dans un souci d'ordre quantitatif grâce à des techniques de simulation.

Toutefois ce système MAESTRO n'est pas intégré à un environnement logiciel complet de spécification et ne résoud donc pas le problème de la duplication entre le travail de spécification et celui d'évaluation. Il s'agit finalement moins d'un environnement de spécification qu'un langage de programmation de haut niveau intégrant des concepts plus appropriés pour représenter le comportement d'un système d'information que les langages classiques. On retrouve également dans [SOL, 1982] une approche fort semblable qui prône l'utilisation d'un langage de programmation de haut niveau comme outil de spécification.

2. Le **système SARA,** proposé dans [RAZOUK,1979] intègre un ensemble de règles et d'outils logiciels pour décrire des systèmes informatiques. La caractéristique essentielle de la démarche proposée par SARA est de s'attacher à montrer que la conception d'un système respecte ses spécifications et est acceptable dans un environnement donné de moyens. Le modèle SARA qui permet de modéliser le comportement distingue également deux aspects dans cette modélisation. La structure de contrôle logique (ou comportement fonctionnel) est exprimée à l'aide d'un formalisme proche des Réseaux de Petri. La description des moyens requis est faite en termes de "processeurs" dont le comportement peut être modélisé par une procédure codée dans un langage de type PL/1.

Un outil de simulation a été développé et permet aussi d'évaluer le comportement des systèmes conçus dans l'environnement SARA. Mais de la même façon que MAESTRO, il manquait dans SARA un environnement logiciel complet de spécification.

Par contre, [WINCHESTER, 1980] a proposé et partiellement réalisé l'intégration de SARA et d'un environnement logiciel défini à l'aide du "Méta Système" proposé par [YAMAMOTO, 1980].

3. Enfin, [SPEWAK, 1981] et [BOYDSTON, 1983] ont conçu et réalisé dans le cadre du projet ISDOS un environnement logiciel compatible avec PSL/PSA intégrant des outils de simulation pour évaluer le comportement de systèmes informatiques.

L'outil logiciel d'évaluation présenté dans ce chapitre se différencie de leurs outils de simulation par

- les modèles retenus, proposés au chapitre 1, plus adaptés pour spécifier le comportement d'un système d'information, en particulier au niveau de ses aspects temporels ;
- les solutions techniques adoptées, offrant une plus grande facilité et une plus grande souplesse d'utilisation, en particulier au niveau de l'exploitation des statistiques.

**Justification d'une approche par simulation**

La simulation informatique adoptée dans le contexte de cet outil d'évaluation est une technique pour **conduire des expériences** à l'aide d'un **programme** exécuté par un ordinateur **sans intervention extérieure** [MAC DOUGALL, 1970] [PRITSKER, 1974] [SCHRIBER, 1974] [VAUCHER, 1971] [ZEIGLER, 1974].

Les spécifications du comportement d'un système d'information représentent l'évolution dans le temps du système spécifié en faisant l'hypothèse que les résultats obtenus sont applicables au comportement réel de ce système.

La **principale raison pour choisir** une approche par simulation réside dans le fait qu'il est **moins coûteux et plus rapide** d'utiliser un programme de simulation, surtout lorsqu'il est automatiquement dérivé, que de réaliser le système, même dans une version prototype, et d'en étudier a posteriori le comportement. De plus, la simulation permet d'étudier ce comportement sous **différentes conditions expérimentales,** en répétant

plusieurs simulations, et donc de faire des analyses statistiques.

Enfin, une approche par simulation offre l'avantage par rapport aux techniques mathématiques plus analytiques

- d'être une **transposition plus immédiate** du comportement puisque les concepts du modèle de spécification peuvent être directement exprimés dans la simulation par une programmation appropriée ;
- de **ne pas nécessiter une traduction complexe,** quand il ne s'agit pas d'une distorsion, des concepts du modèle de spécification sous forme de concepts mathématiques. En effet, les techniques mathématiques disponibles d'évaluation ne permettent pas facilement de prendre en compte tels quels tous les aspects du comportement, en particulier ceux liés au temps et à la synchronisation des processus ;
- et donc de **faciliter** considérablement l'interprétation des résultats produits par l'outil.

**Architecture générale de l'outil d'évaluation**

L'architecture fonctionnelle de cet outil comprend les 3 composants suivants, illustrés à la figure II.1 :

- la GENERATION automatique d'un programme de simulation qui correspond à une version exécutable des spécifications enregistrées dans la base de données de l'environnement logiciel de spécification ;
- l'EXECUTION proprement dite du programme de simulation qui collecte des mesures dans une base de données statistiques ;
- l'EXPLOITATION de cette base de données statistiques, description posthume du comportement simulé, pour produire des rapports de résultats statistiques.

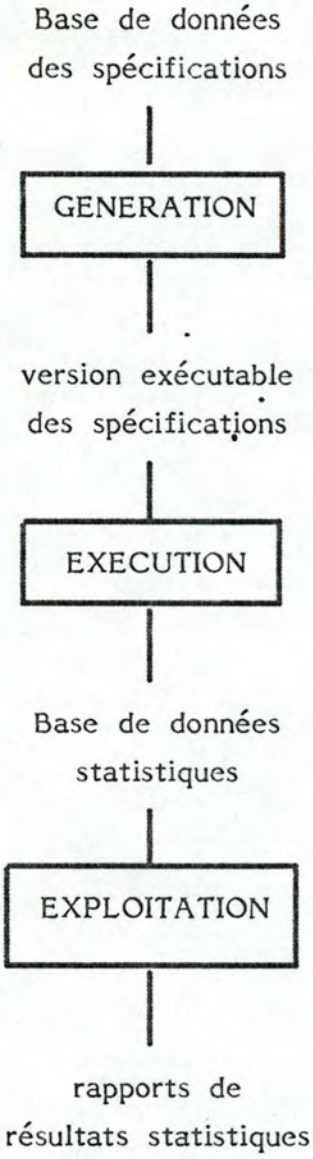



Figure II.1. - Architecture fonctionnelle de l'outil d'évaluation

Les trois sections suivantes présentent respectivement ces 3 composants. Une présentation identique a été adoptée pour les décrire. Premièrement, l'objectif du composant est précisé. Deuxièmement les hypothèses de réalisation retenues sont exposées et les solutions adoptées sont justifiées. Troisièmement, les flux d'informations qui traversent l'outil correspondant sont présentés. Ils constituent les entrées/sorties vues par l'utilisateur et par l'outil lui-même. On distingue, comme illustré à la figure II.2 :

- les paramètres qui prescrivent les options choisies et guident l'exécution ;
- les entrées effectives de l'outil ;
- les sorties effectives de l'outil et
- les rapports qui informent l'utilisateur sur le déroulement de l'exécution et sur ce que l'outil a fait.

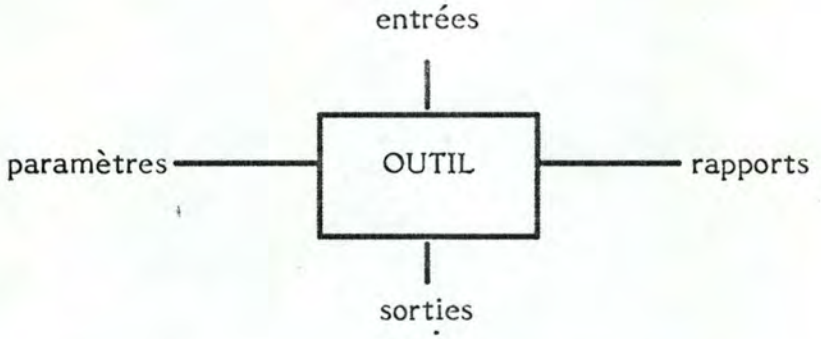



Figure II.2. - Entrées/sorties des outils logiciels

## II.2. GENERATION AUTOMATIQUE D'UN PROGRAMME DE SIMULATION

### II.2.1. Objectif

L'objectif de cet outil de génération est de **dériver automatiquement** un programme de simulation **au départ des spécifications** enregistrées dans la base de données des spécifications.

L'idée sous-jacente à cette génération automatique est de **transformer** des **spécifications déclaratives** et non procédurales en un modèle équivalent mais **exécutable.**

### II.2.2. Solution

Dans ce contexte de transformation de modèles, deux approches sont envisageables. La première revient à directement produire un programme autonome de simulation dans lequel les spécifications ont été traduites sous forme d'instructions du langage de programmation adopté. La seconde solution consiste à utiliser un programme généralisé de simulation qui est réalisé une fois pour toutes. Dans ce cas, les spécifications sont traduites sous forme de tables ou structures de données équivalentes qui guident ou conduisent l'exécution du programme généralisé.

La première approche, qui fut celle choisie pour réaliser les premiers prototypes de cet outil d'évaluation, consiste donc à **générer directement un programme complet et autonome** qu'il suffit de "compiler et linker" pour obtenir une version exécutable. Cependant, pour éviter d'avoir à dériver chaque fois un programme long et complexe car rédigé directement

à l'aide des instructions de base du langage retenu, on peut concevoir et réaliser une machine virtuelle de plus haut niveau, c'est-à-dire un ensemble de primitives pré-programmées. Dans ce cas, le programme à générer est plus simple car il présuppose la disponibilité de cette machine virtuelle et peut donc invoquer ces primitives de plus haut niveau (sous forme d'appels de procédures, par exemple).

Ainsi dans la solution adoptée pour réaliser les premiers prototypes, le langage SIMULA67 [DAHLE, 1970] avait été adopté et la sémantique convenue des (méta)types d'entités et d'associations avait été traduite et figée a priori sous forme respectivement de classes (= types abstraits) et de procédures SIMULA67. La génération se limitait alors à produire du code SIMULA67 étendu à la forme très dépouillée et très proche du langage de spécification.
L'exemple suivant illustre assez les deux formes de spécifications : la première - déclarative - est rédigée à l'aide du langage de spécification tandis que la seconde - exécutable - est rédigée en SIMULA67 étendu.

Exemple II.1.

Forme déclarative :

```
DEF EVENEMENT        arrivée-commande
    DECLENCHE        enregistrement-commande

DEF PROCESSUS        enregistrement-commande
    REQUIERT         1 employé-réception
    ACTIF PENDANT    10 min
    CAUSE            1 commande-à-enregistrer SI com-
                     mande-valide,
                     1 commande-refusée SI commande-
                     invalide
```

Forme exécutable codée - générée automatiquement - en SIMULA67 :

```
EVENEMENT CLASS          arrivée-commande ;
    BEGIN
    DECLENCHE ( ↑ enregistrement-commande,-)
    END ;

PROCESSUS CLASS          enregistrement-commande ;
    BEGIN
    REQUIERT (1, ↑ employé-réception)
    ACTIF-PENDANT ("10 min")
    CAUSE (1,↑ commande-à-enregistrer, SI (↑ commande-valide))
    CAUSE (1,↑ commande-refusée, SI (↑ commande-invalide))
    END ;
```

On notera la grande similitude entre les deux formes de la même spécification. La principale différence réside dans les particularités syntaxiques des deux langages, en particulier au niveau de l'ordre des instructions qui est plus rigide en SIMULA67. Cette similitude est rendue possible parce que certaines classes et procédures ont été écrites a priori - indépendamment de toute spécification. L'exemple ci-dessus suppose en effet que les classes EVENEMENT et PROCESSUS ainsi que les procédures DECLENCHE, REQUIERT, ACTIF-PENDANT, CAUSE et SI aient .été codées en SIMULA67 avant toute "génération".

La programmation, par exemple, de la classe SIMULA67 PROCESSUS traduit le comportement convenu de tout processus ; en particulier, elle précise, sous forme de procédure que les seules actions possibles d'un processus sont :

- REQUIERT (taux, ressource) pour créer une requête ;
- ACTIF-PENDANT (durée) pour devenir actif, pendant une certaine durée, dès que les ressources requises sont disponibles ;
- CAUSE (cardinalité, événement, condition) pour éventuellement causer la survenance de certains événements au terme de sa durée d'exécution.

L'attrait de cette approche tient essentiellement à la **"continuïté d'écriture"** entre les spécifications et le programme qui les traduit sous forme exécutable. Même si cette approche n'a finalement pas été retenue dans la version finale et opérationnelle de l'outil, il semble cependant qu'elle offre un intérêt certain dans le contexte général de production automatique de code. En effet, elle fournit une grande transparence à l'utilisateur qui retrouve dans le texte généré un formalisme très proche de celui qu'il avait utilisé pour rédiger les spécifications. L'intérêt de cette "continuïté d'écriture" est d'autant plus grand que l'utilisateur pourrait être amené à modifier ou adapter manuellement le code produit.

A contrario, cette dernière possibilité n'étant pas requise dans l'outil d'évaluation envisagé, puisqu'on souhaitait que les modifications ne puissent nécessairement se faire qu'au niveau des spécifications, il a semblé plus approprié de **choisir** la seconde approche de **génération par tables.**

Cette seconde technique consiste donc à **ne réaliser qu'un seul programme** généralisé, écrit une fois pour toutes, dont **l'exécution sera conduite par des tables.** Dans ce cas, les spécifications sont extraites de la

base de données et tranformées en une structure de données (ou tables) équivalente et reconnue par le programme généralisé.

Cette solution offrait l'avantage, dans le contexte de cet outil d'évaluation, de rendre impossible toute modification manuelle de la version exécutable des spécifications et donc de garantir que celle-ci était bien équivalente aux spécifications enregistrées dans la base de données. Cette solution impose en effet que tout changement dans les spécifications soit effectué à l'aide du langage de spécification pour que leur incidence soit analysée par l'outil d'évaluation.

Conceptuellement, cette étape de génération préalable à une simulation n'est donc pas indispensable car on pourrait imaginer que la base de données corresponde en fait directement aux tables du simulateur. Le maintien de cette étape de génération ne se justifie essentiellement que par des raisons de performances : la structure générée est rigoureusement équivalente à celle contenue dans la base de données mais elle est optimalisée en vue de la simulation. De plus, cet outil de génération effectue certains contrôles pour s'assurer que la simulation souhaitée est possible au départ des spécifications telles qu'elles sont enregistrées dans la base de données.

### II.2.3. Entrées-Sorties

Les seuls **paramètres** à préciser lorsqu'on invoque cet outil de génération servent à déterminer et **sélectionner** le ou les **schémas** qu'on souhaite évaluer et donc simuler.

Les **entrées** de l'outil sont constituées uniquement par les **spécifications** enregistrées dans la **base de données** dans laquelle l'outil peut retrouver automatiquement au départ, par exemple, du seul nom d'un schéma toutes les classes qui caractérisent le comportement spécifié.

Les spécifications extraites de la base de données sont alors transformées sous forme de **tables** équivalentes qui constituent les **sorties** de l'outil et serviront à guider l'exécution de la simulation.

En plus de cette transformation en tables, l'outil de génération effectue un certains nombres de **contrôles** pour s'assurer que la simulation

souhaitée est possible. Ces contrôles correspondent en fait à certains des contrôles de complétude et de cohérence repris et développés dans le chapitre suivant. Un **rapport** signale donc les anomalies détectées qui subsistent dans la base de données et empêchent l'exécution de la simulation correspondante. En particulier, il peut détecter et signaler que certains éléments de la spécification manquent (cf. III.2.) tels que :

- l'échéancier de certaines classes d'événements initiaux ;
- le prédicat de certaines conditions de déclenchement ;
- la probabilité associée à certaines conditions de survenance ;
- le domaine des valeurs - possibles - de certains attributs ou paramètres.

Il peut également détecter certaines incohérences dans les spécifications (cf. III.3.) et signaler que

- la durée d'exécution de certains processus NON interruptibles est supérieure à l'intersection des périodes les plus larges du calendrier de disponibilité des ressources requises ;
- un taux de réquisition est supérieur à la capacité maximum disponible d'une ressource.

Les autres contrôles de cohérence qui seront développés au chapitre suivant n'empêchent pas de lancer une simulation et ne font donc l'objet d'aucune vérification systématique et préalable à une simulation. Par contre, s'ils n'ont pas été vérifiés préalablement, la simulation restituera évidemment un comportement incohérent et non représentatif de la réalité.

## II.3. EXECUTION DE LA SIMULATION

### II.3.1. Objectif

L'objectif de cet outil est de simuler le comportement spécifié sur une période de temps donnée et de collecter des mesures sur le fonctionnement simulé.

### II.3.2. Solution

La technique adoptée est une **simulation à événements discrets** [FISHMAN, 1973] [FRANTA, 1977] dont l'exécution est **conduite par les tables** produites par l'outil de génération automatique. Cette approche par événements discrets convient mieux qu'une technique de simulation continue pour prendre en considération les phénomènes essentiellement discrets du

comportement d'un système d'information. Une simulation continue offre en effet une vision de flux d'informations continus dans le temps où les changements d'état s'opèrent de façon continue et sont, d'ailleurs, souvent modélisés par des équations différentielles [FORRESTER, 1961]. Il est dès lors assez malaisé de traduire les situations conventionnelles de décision et de synchronisation caractéristiques du comportement d'un système d'information (cf. I.2.6.). De plus, il est difficile de représenter et donc d'évaluer les effets des blocages et autres synchronisations dus à l'indisponibilité temporaire de certaines ressources. C'est pourquoi l'approche par événements discrets adoptée a semblé plus proche de la vision introduite dans le modèle de spécification proposé au chapitre 1.

La **simulation** à événements discrets est **fondée sur la gestion d'un échéancier** [FRANTA, 1977] [VAUCHER, 1975] [WYMAN, 1975] ou liste d'événements planifiés, ordonnés par dates de survenance croissantes. L'exécution de la simulation est alors dynamiquement guidée par cet échéancier dont les événements sont exploités dans l'ordre croissant de leur date de survenance. L'exploitation d'un événement revient à ajuster l'horloge de la simulation - ou temps courant simulé - à la date de survenance de l'événement rencontré - ou événement courant - et d'exécuter certaines actions associées à la survenance de l'événement. Certaines de ces actions peuvent à leur tour modifier l'échéancier par l'insertion ou la suppression d'autres événements planifiés. Les événements pouvant intervenir dans la simulation, et donc être planifiés dans l'échéancier, appartiennent à des types prédéfinis auxquels a été associée une procédure ou ensemble d'actions susceptibles de modifier l'état du système simulé, y compris son échéancier. Quand un événement de l'échéancier devient courant, son type est examiné et la procédure correspondante est exécutée.

Les types d'événements - planifiables dans l'échéancier de la simulation - traduisent les visions "Evénement-Processus" et "Processus-Ressource" introduites dans le modèle de spécification adopté. Dans ce modèle, on considère un système d'information comme une collection de processus créés (déclenchés) dynamiquement en réponse à la survenance d'événements et concurrents pour l'obtention de ressources partagées. L'outil de simulation intègre donc dans un programme généralisé, appelé à être conduit par tables, la sémantique convenue des types d'entités et d'associations du modèle présenté au chapitre 1.

**Composants de l'outil logiciel d'exécution**

L'outil logiciel d'exécution s'articule autour des composants illustrés à la figure II.3.

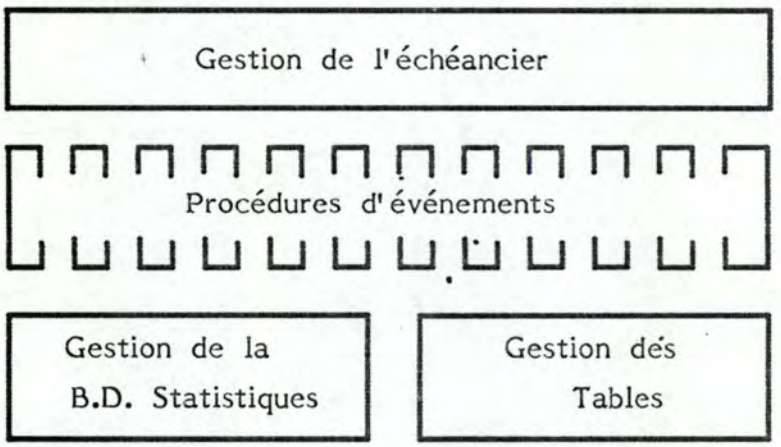



Figure II.3. - Structure de l'outil de simulation

A. Le **gestionnaire de l'échéancier** est le "moteur" de la simulation, sa fonction est de

- sélectionner l'événement en tête de l'échéancier, c'est-à-dire celui dont la date de survenance est la plus rapprochée. Cet événement devient l'événement courant de la simulation ;
- ajuster l'horloge de la simulation à la date de survenance de l'événement courant. Cette date devient le temps-simulé-courant ;
- examiner le type de l'événement et invoquer la procédure d'événement correspondante.

B. Le **gestionnaire des tables** est constitué d'un ensemble de primitives qui permettent de retrouver les éléments de spécification générés par l'outil de génération (cf. II.1.). Ces tables sont équivalentes aux spécifications du comportement qu'on souhaite simuler et évaluer.

C. Le **gestionnaire de la base de données statistiques** est un interface par lequel passent tous les accès à la base de données qui regroupe les mesures prises pendant la simulation. Chaque survenance d'un événement, c'est-à-dire lorsqu'il devient courant ou en tête de l'échéancier, donne lieu à une mesure reprenant notamment :

- le type d'événement (ex. déclenchement d'un processus) ;
- le nom de l'entité concernée (ex. processus "Enregistrement de commande") ;
- un identifiant interne d'occurrence (ex. "Enregistrement de commande" 1234) ;
- et la date de survenance de l'événement (ex. 1er jour 8H 30 min).

En fonction du type d'événement et du type d'entité, cette dernière date peut en fait correspondre, dans le modèle présenté au chapitre 1 :

- soit à une date de survenance d'un événement ;
- soit à une date de déclenchement, de démarrage, d'interruption, de redémarrage ou de terminaison d'un processus ;
- soit encore à une date de réquisition, d'acquisition ou de libération d'une ressource par un processus.

Ces mesures sont enregistrées dans la base de données de façon à permettre la production de rapports statistiques à l'aide des outils d'exploitation ad hoc (cf. III.3.).

D. Les **procédures d'événements** traduisent la sémantique CONVENUE des concepts du modèle exposé au chapitre 1. Ces procédures constituent la définition opérationnelle (= programmée) des différentes réactions aux **événements répertoriés** de la simulation, c'est-à-dire ceux qui peuvent être planifiés dans l'échéancier :

1. **Début de la simulation**

Cet événement est automatiquement planifié dans l'échéancier lorsqu'on invoque l'outil de simulation et la procédure qui lui est associée correspond en fait à l'initialisation de l'exécution.

La première action correspond à la prise en compte des spécifications qui précisent la **disponibilité** des différentes ressources concernées par le schéma à évaluer. Cette prise en compte se traduit, pour chaque ressource, par la planification dans l'échéancier d'un événement "changement de capacité maximum d'une ressource" à une date - simulée - correspondant au début de la première période de disponibilité de la ressource.

La seconde action revient à prendre en compte les spécifications qui précisent l'**échéancier** des différents types d'événements initiaux du schéma à simuler. Cette prise en compte se traduit, pour chaque type d'événements initiaux, par la planification dans l'échéancier de la simulation d'un ou plusieurs événements "survenance d'un événement" à une date - simulée - correspondant à la première date de l'échéancier spécifié.

Enfin, la troisième action est la planification immédiate ( au temps

simulé courant = 0), d'un événement "acquisition des paramètres".

**2. Acquisition des paramètres**

Le fait de considérer l'acquisition des paramètres comme une procédure d'événement offre la possibilité à l'utilisateur d'introduire des **points d'arrêt** dans la simulation. Ces points d'arrêt peuvent alors être mis à profit pour visualiser certaines variables d'état ou certaines files d'événements (échéancier ou file d'attente).

La première action lorsqu'un tel événement devient courant se traduit par la planification de l'événement "fin de la simulation" à la date - simulée - fixée comme paramètre par l'utilisateur.

La seconde action correspond à l'éventuelle visualisation - à la demande - de certaines variables d'état (capacité disponible d'une ressource, nombre de processus en attente, ...) et de l'échéancier de la simulation.

Le dernière action revient à planifier un autre événement "acquisition des paramètres" à une date ultérieure - simulée - donnée par l'utilisateur.

A part ces éventuelles acquisitions de paramètres, l'exécution de la simulation se déroule sans intervention de l'utilisateur.

**3. Fin de la simulation**

Cet événement lorsqu'il devient courant signale l'arrêt de la simulation.

**4. Changement de capacité maximale d'une ressource**

Ce type d'événement signale un changement de la capacité maximale disponible d'une ressource en fonction de son calendrier de disponibilité.

La première action à exécuter lorsqu'un tel événement devient courant consiste à **ajuster la nouvelle capacité** maximale en examinant dans

les tables la spécification correspondante. De plus, lorsque la capacité maximale diminue (devient éventuellement nulle), cet ajustement peut entraîner la planification immédiate de certaines "interruptions de processus" qui utilisaient cette ressource. Par contre, lorsque la capacité maximale augmente, cet ajustement peut alors amener la planification immédiate de certains "(re)démarrages de processus" qui étaient en attente de cette ressource.

La seconde action revient à prendre en compte les spécifications qui précisent le **calendrier de disponibilité** de la ressource pour planifier un nouveau "changement de capacité maximale" à une date - simulée - correspondant au prochain changement spécifié.

**5. Survenance d'un événement**

La procédure associée à ce type d'événement traduit la sémantique convenue des spécifications de **déclenchement** définies selon le modèle proposé au chapitre 1. La prise en compte de ces spécifications peut entraîner pour chaque déclenchement possible au départ de cet événement

- soit la planification immédiate (c'est-à-dire au même temps simulé) d'un ou plusieurs événements "déclenchement d'un processus" si les modalités de déclenchement sont vérifiées ;
- soit la mémorisation de l'événement en vue d'une contribution ultérieure à un déclenchement si les modalités de déclenchement ne sont pas vérifiées du fait de sa survenance.

Si un délai de temporisation est spécifié dans les modalités de déclenchement (cf.I.2.5.), un événement "survenance d'un événement" (le même) sera alors replanifié dans l'échéancier à une date correspondant à la date courante plus le délai précisé.

Enfin, si l'événement appartient à un type d'événement initial du schéma, il convient également d'exploiter les spécifications qui précisent l'**échéancier** du type d'événement concerné pour planifier un ou plusieurs événements "survenance d'un événement" à une date - simulée - correspondant à la date suivante spécifiée dans l'échéancier du type d'événement.

**6. Déclenchement d'un processus**

La procédure associée à ce type d'événement traduit la

sémantique convenue du comportement des processus, tel que précisé dans le modèle exposé au chapitre 1. (cf. I.3.5.) :

- un processus déclenché ne démarre réellement qu'une fois en possession de toutes les ressources qu'il requiert directement ou via d'autres ressources ; aucune allocation partielle (ou réservation) n'est effectuée ;
- de plus, le processus requiert la portion des ressources pendant toute sa durée d'exécution.

L'action principale à effectuer lorsqu'un tel événement devient courant consiste donc à prendre en compte les spécifications qui précisent les **réquisitions** de ce type de processus pour vérifier si toutes les ressources requises sont disponibles.
Une ressource est disponible lorsque la quantité requise est inférieure ou égale à la capacité disponible. De plus, si le processus est NON interruptible, il faut que la quantité de ressource requise puisse lui rester allouée sans interruption pour toute la durée d'exécution spécifiée du processus. Enfin, si la ressource est indisponible parce qu'utilisée par des processus interruptibles et moins prioritaires, certaines "interruptions de processus" peuvent devoir être planifiées et exécutées immédiatement pour libérer (préemption) une partie de la ressource.

En fonction de ces vérifications,

- si toutes les ressources sont disponibles, un "démarrage de processus" peut être planifié au même temps simulé ;
- par contre, si une partie des ressources requises reste indisponible, ce "démarrage de processus" est mis en attente d'une future disponibilité des ressources manquantes. Ces ressources deviendront disponibles en fonction des "terminaisons de processus" ou "changements de capacité maximum" ultérieurs.

### 7. (Re)démarrage d'un processus

Lorsqu'un tel événement devient courant, on est sûr que toutes les ressources qu'il requérait sont disponibles. Dès lors, il s'agit principalement de prendre en compte les spécifications qui précisent les **réquisitions** de ce processus en vue d'**allouer** les ressources requises.
Cette allocation d'une certaine quantité de ressources se traduit au niveau de la ressource par une diminution de sa capacité disponible d'un montant équivalent à la quantité requise.

Enfin, une dernière action consiste à exploiter la spécification de la **durée d'exécution** du processus pour planifier dans l'échéancier une "terminaison de processus" à une date - simulée - correspondant à la date courante plus la durée d'exécution précisée.

**8. Terminaison d'un processus**

Lorsqu'un événement de ce type devient courant, il s'agit principalement de prendre en compte les spécifications qui précisent les **réquisitions** de ce processus en vue de **libérer** les ressources utilisées. Cette libération se traduit pour une ressource par une augmentation de sa capacité disponible d'un montant équivalent à la quantité requise et donc utilisée par le processus qui se termine. De plus, cette augmentation de la capacité disponible peut entraîner la planification immédiate d'autres "(re)démarrages de processus" qui étaient en attente de cette ressource.

Enfin, une dernière action doit être effectuée pour prendre en compte les spécifications qui définissent les **survenances** d'événements du schéma causées par ce processus. Cette prise en compte se traduit par la planification d'un certain nombre d'événements "survenance d'un événement", suivant les précisions contenues dans les modalités de survenance (cf. I.2.4.).

**9. Interruption d'un processus**

Ce type d'événement qui peut être dû à la survenance d'un événement "déclenchement d'un processus" plus prioritaire ou "changement de capacité maximale d'une ressource" entraîne les mêmes effets qu'une "terminaison de processus". Mais en plus, il veille à :

- annuler de l'échéancier l'événement "terminaison du processus" correspondant ;
- ajuster la nouvelle durée d'exécution - amputée du temps pendant lequel le processus a déjà été actif - et enfin
- mettre en attente un événement "redémarrage du processus".

Ces 9 procédures d'événements, associées aux tables générées qui représentent les spécifications, déterminent complètement le comportement du ou des schémas à simuler. De plus, chaque fois qu'un tel événement survient, il engendre automatiquement une mesure qui est transmise au

gestionnaire de la base de données statistiques afin d'y être enregistrée.

### II.3.3. Entrées-Sorties

Les **paramètres** de l'outil d'exécution servent à définir
la **durée de la simulation**
et le **type de renseignements** fournis par le simulateur en cours d'exécution.

Ces paramètres présentent la particularité de pouvoir être modifiés en cours de simulation par le placement de points d'arrêt dans l'exécution à des dates - simulées - données. Ces points d'arrêt offrent la possibilité à l'utilisateur d'interrompre la simulation et de visualiser certaines files d'attente et variables d'état. Sur base des renseignements obtenus, les deux paramètres mentionnés ci-dessous peuvent être modifiés avant de redémarrer la simulation.

Les seules **entrées** de cet outil sont constituées par les **tables produites** par l'outil de génération automatique et qui guident l'exécution de la simulation.

Aucun rapport statistique n'est produit par l'outil d'exécution. Par contre, toutes les mesures prises en cours d'exécution sont enregistrées dans une **base de données statistiques** qui constitue la **sortie** de l'outil. Cette base de données est en fait une description posthume du comportement simulé.

Si aucune exploitation statistique n'est réalisée en cours d'exécution, un **rapport** peut cependant être produit à la demande. Il s'agit d'une **trace** qui reprend chronologiquement tous les événements détectés en cours de simulation lorsqu'ils sont devenus courants. Une ligne de cette trace reprend pour un événement les éléments de la mesure enregistrée dans la base de données lorsqu'il est devenu courant et notamment :

- la date de survenance ;
- le type d'événement ;
- le nom de l'entité concernée ;
- un identifiant interne d'occurrence.

Ce rapport peut en plus fournir des explications et des justifications plus ou moins complètes sur ce qui s'est réellement passé dans la simulation. Il peut par exemple lorsqu'un événement du schéma survient, préciser quel autre événement manque et empêche donc le déclenchement d'un processus ou encore signaler lorsqu'un processus est déclenché, quelle ressource est indisponible et empêche le (re)démarrage.

## II.4. EXPLOITATION DES RESULTATS STATISTIQUES DE LA SIMULATION

### II.4.1. Objectif

L'objectif de cet outil est d'extraire certaines mesures élémentaires de la base de données statistiques et de les traiter pour produire des **résultats statistiques.** Ces résultats statistiques doivent permettre l'analyse des performances obtenues par la simulation et leur comparaison avec celles attendues par l'utilisateur. L'approche retenue n'est donc pas de fournir à l'utilisateur des outils d'analyses statistiques (construction d'intervalles de confiance, analyse de séries chronologiques, détection de l'état stationnaire...) mais de mettre à sa disposition des outils, **restituant le comportement spécifié** sous forme de **statistiques descriptives.** Il incombera dès lors à l'utilisateur de mener lui-même la stratégie d'analyse des résultats statistiques la plus appropriée sachant qu'il dispose des matériaux de base pour l'aider dans sa démarche.

### II.4.2. Solution

Dans une approche par simulation informatique, deux solutions sont envisageables pour la production de résultats statistiques :

- le programme de simulation peut produire lui-même directement des rapports statistiques en fonction des paramètres fournis **avant** l'exécution de la simulation ;
- le programme peut collecter, en cours d'exécution, des mesures dans une base de données qui constitue en fait la description posthume du comportement simulé. Cette base de données peut **ensuite** être exploitée à l'aide d'outils logiciels appropriés pour restituer des images statistiques globales ou partielles de ce comportement simulé.

Cette seconde solution, qui a été retenue dans la version finale de l'outil développé, offre l'avantage de pouvoir produire des rapports

statistiques à la demande et a posteriori sans avoir à réexécuter une simulation. Elle favorise donc une exploitation progressive des résultats et autorise différentes stratégies d'exploitation des résultats statistiques. Cette solution semble, en effet, plus souple et en particulier mieux prendre en compte les considérations pratiques suivantes :

- c'est souvent l'étude de résultats statistiques qui suscite l'intérêt pour d'autres résultats - non initialement prévus - pour tirer tous les enseignements d'une simulation ;
- dans une solution intégrée où les résultats sont directement produits par le programme de simulation, on est tenu de relancer une nouvelle simulation - toujours gourmande en temps d'exécution - si l'on souhaite obtenir d'autres résultats - non initialement demandés - sur le même comportement simulé.

Les outils logiciels d'exploitation permettent donc de produire à la demande, au départ de la base de données statistiques et sous différentes formes, des résultats statistiques caractérisant le comportement d'une ou plusieurs classes d'objets sur une période donnée.
Ces résultats statistiques concernent les aspects suivants du comportement :

1. survenance des événements (cf. I.2.3.) ;
2. déclenchement des processus (cf. I.2.4.) ;
3. évolution - temporelle - des processus (cf. I.2.5.) ;
4. comportement des ressources (cf. I.2.5.).

Les résultats statistiques qui apparaissent dans les rapports produits se présentent principalement sous la forme de :

a) **nombres** de changements d'état, caractéristiques du comportement des objets considérés (exemple : nombre de déclenchements) ;
b) **durées** de certains états caractéristiques par lesquels évoluent les objets considérés (exemple : durée d'attente d'un processus) ;
c) **volumes** ou nombres d'objets dans certains états caractéristiques à un moment donné (exemples : nombre de processus en attente à l'instant t ou capacité disponible d'une ressource à l'instant t).

Les résultats statistiques concernant des durées et des volumes mettent systématiquement en évidence les valeurs minimales, moyennes, maximales et écart-type de la distribution de probabilité correspondante. Toutefois et à titre d'illustration, seule l'expression analytique de la valeur moyenne sera présentée dans l'exposé des résultats statistiques ci-dessous.

Enfin, tout résultat statistique est donné pour une période de référence qui est fixée en précisant sa date de début et sa date de fin. Cela signifie qu'elle peut ne pas nécessairement correspondre à la durée simulée. Cela permet dès lors de ne s'intéresser qu'à certaines périodes "critiques" de la période simulée et, par exemple, de ne pas prendre en compte certaines mesures correspondant à la "période de chauffe" du modèle - avant que le système n'ait atteint son état d'équilibre. Cela permet également de présenter sur un même rapport des résultats statistiques concernant différentes périodes et de dégager l'évolution d'un phénomène. Par contre, le soin de choisir les périodes considérées comme statistiquement significatives reste du ressort de l'utilisateur et aucun outil spécialisé ne lui est actuellement fourni pour tester, par exemple, si le système est stationnaire - au sens statistique (moyenne et variance constantes) [FICHEFET, 1976] [KOBAYASHI, 1978].

### II.4.2.1. Résultats statistiques concernant la survenance des événements

Le comportement des événements d'une classe E, sur une période T donnée, peut être caractérisé par les résultats statistiques suivants :

a) le **nombre de survenances** NSUR (E, T) d'événements de la classe E dans la période T considérée. Ce résultat est obtenu par simple comptage et correspond en fait à un nombre d'observations puisqu'une mesure élémentaire est prise à chaque survenance d'événement pendant la simulation ;

b) pour chaque classe d'événements prédécesseurs (successeurs) des événements de la classe E survenus dans la période T, la **durée inter-événements** sachant que

- une durée **inter-événements** DIE ($e_i$, $ae_j$) est la durée comprise entre la date de survenance d'un événement $e_i$ de la classe E et celle d'un autre événement $ae_j$ d'une classe AE qui le précède (lui succède),

  par exemple, la durée comprise entre la survenance d'un événement "commande du jour" et celle de l'événement "commande expédiée" correspondant ;

- les mesures prises pendant la simulation permettent de reconstituer, pour la période T considérée, la distribution de probabilité correspondante dont la valeur moyenne, par exemple, peut être déterminée par :

$$\frac{1}{NSUR(E,T)} \sum_{i=1}^{NSUR(E,T)} \frac{1}{NPS(e_i,AE)} \sum_{j=1}^{NPS(e_i,AE)} DIE(e_i,ae_j)$$

où NPS ($e_i$, AE) est le nombre d'événements de la classe AE qui précèdent (succèdent à) l'événement $e_i$.

### II.4.2.2. Résultats statistiques concernant le déclenchement des processus

L'évolution des déclenchements de processus d'une classe P, sur une période T donnée, peut être caractérisée par les résultats statistiques suivants :

a) le **nombre de déclenchements** NDEC (P,T) de processus de la classe P dans la période T considérée ;

b.1) la **durée de déclenchement** des processus de la classe P déclenchés dans la période T sachant que :

- une durée de déclenchement DDEC ($p_i$) est la durée comprise entre la date de déclenchement d'un processus $p_i$ de la classe P et la date de survenance de l'événement le plus ancien ayant contribué à son déclenchement ;
- la valeur moyenne de la distribution de probabilité correspondante peut être déterminée par

$$\frac{1}{NDEC(P,T)} \sum_{i=1}^{NDEC(P,T)} DDEC(p_i)$$

b.2) pour chaque classe d'événements EC contribuant au déclenchement de processus de la classe P déclenchés dans la période T, la **durée de contribution** sachant que

- une durée de contribution DCONT ($p_i$, $ec_j$) est la durée comprise entre la date de déclenchement d'un processus $p_i$ de la classe P et la date de survenance d'un événement $ec_j$ de la classe EC ayant contribué à son déclenchement ;
- la valeur moyenne de la distribution correspondante peut être déterminée par

$$\frac{1}{NDEC(P,T)} \sum_{i=1}^{NDEC(P,T)} \frac{1}{NC(p_i,EC)} \sum_{j=1}^{NC(p_i,EC)} DCONT(p_i,ec_j)$$

où NC ($p_i$, EC) est le nombre d'événements de la classe EC ayant contribué au déclenchement du processus $p_i$.

### II.4.2.3. Résultats statistiques concernant l'évolution des processus (cf. I.3.5)

Le comportement - ou évolution temporelle - des processus d'une classe P, sur une période T donnée, peut être caractérisé par les résultats statistiques suivants :

a) les **nombres de déclenchements** NDEC (P,T), de **démarrages** NDEM (P,T), d'**interruptions** NINT (P,T), de **redémarrages** NRED (P,T) et de **terminaisons** NTER (P,T) de processus de la classe P dans la période T considérée. Chacun de ces résultats correspond en fait à un nombre d'observations puisque chaque changement d'état d'un processus donne lieu à une mesure dans la base de données statistiques de la simulation ;

b) les **durées d'attente, d'interruption, d'activité** et **totale d'exécution** des processus de la classe P terminés dans la période T sachant que :
   - une durée d'attente DATT ($p_i$) est la durée comprise entre la date de déclenchement et celle de démarrage d'un processus $p_i$ ;
   - une durée d'interruption DINT ($p_i$) est la durée pendant laquelle un processus $p_i$ qui a été actif est interrompu et en attente de ressources. Puisqu'un processus peut être interrompu plusieurs fois, elle correspond à la somme des durées d'**une** interruption du processus $p_i$ où la durée d'**une** interruption DIINT (j, $p_i$) représente la durée comprise entre la date de la jème interruption du processus $p_i$ et celle du redémarrage correspondant ;
   - une durée d'activité DACT ($p_i$) est la durée pendant laquelle un processus $p_i$ est actif et supposé exécuter les règles de sa procédure. Les résultats statistiques associés ne sont donc qu'une restitution des spécifications de durée (cf. I.3.2.) ;
   - une durée totale d'exécution DEXE ($p_i$) est la durée comprise entre les dates de déclenchement et de terminaison d'un processus $p_i$. Elle correspond donc à la somme des durées d'attente, d'interruption et d'activité d'un processus $p_i$ ;
   - la valeur moyenne de la distribution de probabilité de ces durées peut être déterminée par :

$$\frac{1}{NTER(P,T)} \sum_{i=1}^{NTER(P,T)} \text{Durée } (p_i)$$

où Durée $(p_i)$ représente respectivement DATT $(p_i)$, DINT $(p_i)$, DACT $(p_i)$ et DEXE $(p_i)$,

c) les **volumes** de processus de la classe P **en attente, interrompus, actifs** et **en cours d'exécution** à un moment donné, sachant que :

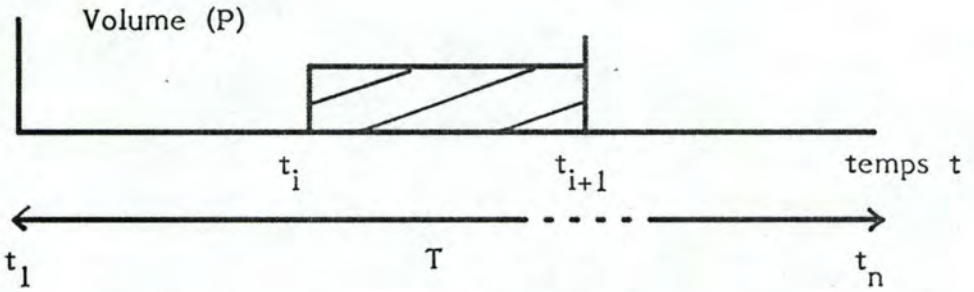
- un volume "en attente" VATT (P,t) représente le nombre de processus de la classe P en attente de ressources au temps t. Puisqu'un processus est en attente s'il a été déclenché mais n'a pas encore pu démarrer, ce volume correspond à la différence entre les nombres de déclenchements NDEC (P,T) et de démarrages NDEM (P,T) pour la période T définie par les bornes O et t ;
- un volume "en interruption" VINT (P,t) représente le nombre de processus de la classe P interrompus au temps t. Puisqu'un processus est interrompu s'il a été interrompu mais n'a pas encore pu redémarrer, ce volume correspond à la différence entre les nombres d'interruptions NINT (P,T) et de redémarrages NRED (P,T) pour la période T définie par les bornes O et t ;
- un volume "en activité" VACT (P,t) représente le nombre de processus de la classe P actifs au temps t. Puisqu'un processus est actif s'il a démarré ou redémarré mais n'a pas été interrompu ou n'est pas terminé, ce volume correspond à
  (NDEM (P,T) + NRED (P,T) - (NINT (P,T) + NTER (P,T)) pour la période T définie par les bornes O et t ;
- un volume "en cours d'exécution" VEXE (P,t) représente le nombre de processus en cours d'exécution au temps t. Puisqu'un processus est en cours d'exécution s'il a été déclenché mais n'est pas encore terminé, ce volume correspond à la différence entre les nombres de déclenchements NDEC (P,T) et de terminaisons NTER (P,T) pour la période T définie par les bornes O et t. Ce volume est la somme des trois précédents ;
- la valeur moyenne de la distribution de probabilité de ces volumes peut être déterminée par :

$$\frac{1}{T} \int_0^t \text{volume } (P,t)dt$$

où volume (P,t) représente respectivement VATT (P,t), VINT (P,t), VACT (P,t) et VEXE (P,t).

La solution adoptée qui consiste à prendre une mesure à chaque changement d'état d'un processus permet de restituer une valeur exacte de ces volumes moyens puisque le calcul de l'intégrale ne fait l'objet d'aucune approximation. Sachant, en effet, que le volume (P,t) ne peut varier qu'à l'occasion d'un changement d'état, donc d'une mesure, on peut facilement calculer le volume moyen sur la période T considérée :

- par une reconstitution de la fonction "en escalier" Volume (P,t) illustrée à la figure suivante :




- et l'application de la formule de calcul suivante :

$$\frac{1}{t_n - t_1} \sum_{i=1}^{n} \text{Volume } (P,t_i) * (t_{i+1} - t_i)$$

où $t_i$ est la date du ième changement d'état - donc ième mesure - de processus de la classe P dans la période T considérée ;

$t_1$ est la date du 1er changement d'état dans la période ;

$t_n$ est la date du dernier changement d'état dans la période et volume $(P,t_i)$ est le volume constaté à l'occasion du ième changement d'état.

Par rapport à une solution qui consiste à prendre des mesures dans la simulation de façon régulièrement espacée dans le temps (toutes les minutes simulées, par exemple), cette solution offre l'avantage de parer au danger que la variation de volume soit également périodique et en phase avec le processus d'échantillonnage périodique.

Au départ des résultats statistiques mentionnés ci-dessus, on peut imaginer différentes variantes. On peut, par exemple, vouloir déterminer la

durée moyenne d'une interruption des processus d'une classe P par l'application de la formule suivante :

$$\frac{1}{NINT(P,T)} \sum_{j=1}^{NINT(P,T)} DIINT(j,P)$$

où DIINT (j,P) est la durée de la jème interruption de processus de la classe P.

Puisque la durée d'attente DATT$^{\cdot}$ ($p_i$) est définitivement connue lorsque le processus $p_i$ a démarré, on peut préférer calculer la durée moyenne d'attente pour les processus de la classe P **déclenchés plutôt que terminés** dans la période T considérée et appliquer la formule suivante :

$$\frac{1}{NDEM(P,T)} \sum_{i=1}^{NDEM(P,T)} DATT(p_i)$$

On peut également souhaiter distinguer la cause de l'attente ou de l'interruption d'un processus et extraire de ses durées d'attente et d'interruption le temps pendant lequel la capacité maximale disponible d'une ressource requise est nulle en fonction de son calendrier de disponibilité. L'extraction et la mise en évidence de ce temps "**mort**" permet de ne considérer comme temps d'attente et d'interruption que les temps pendant lesquels un processus est bloqué et en attente de ressources indisponibles parce que réellement utilisées par d'autres processus au même moment. Cette façon de procéder présente l'avantage de

- restituer des durées d'attente et d'interruption significatives dans les intervalles de temps pendant lesquels les ressources sont effectivement disponibles (journées de travail, par exemple) ;
- ne pas comptabiliser les intervalles de temps pendant lesquels les ressources sont de toute manière indisponibles pour qui que ce soit (nuits et week-end, par exemple).

### II.4.2.4. Résultats statistiques concernant le comportement des ressources

Le comportement d'une ressource R, sur une période T donnée, est principalement caractérisé par les résultats statistiques suivants :

a) les **nombres de réquisitions** NREQ (R,T), **d'acquisitions** NACQ (R,T) et **de libérations** NLIB (R,T) d'une portion de la ressource R dans la période T considérée ;

b) les **durées d'attente, d'utilisation** et **totale de service** des requêtes, portant sur la ressource R, ayant fait l'objet d'une libération dans la période T considérée, sachant que :
   - une durée d'attente DREQ ($r_i$) est la durée comprise entre les dates de réquisition et d'acquisition d'une requête $r_i$ portant sur la ressource R ;
   - une durée d'utilisation DUTI ($r_i$) est la durée comprise entre les dates d'acquisition et de libération d'une requête $r_i$ ;
   - une durée totale de service DSER ($r_i$) est la durée comprise entre les dates de réquisition et de libération d'une requête $r_i$ et correspond donc à la somme des durées d'attente DREQ ($r_i$) et d'utilisation DUTI ($r_i$) ;
   - la valeur moyenne de la distribution de probabilité de ces durées peut être déterminée par :

$$\frac{1}{NLIB(R,T)} \sum_{i=1}^{NLIB(R,T)} Durée\ (r_i)$$

   où Durée ($r_i$) représente respectivement DREQ ($r_i$), DUTI ($r_i$) et DSER ($r_i$) ;

c.1) les **volumes** de requêtes, portant sur la ressource R, **en attente, en cours d'utilisation** et **en cours de service** à un moment donné, sachant que :
   - un volume "en attente" VREQ (R,t) représente le nombre de requêtes pour lesquelles la portion de ressource R a été requise mais pas encore acquise au temps t. Ce volume correspond donc à la différence entre les nombres de réquisitions NREQ (R,T) et d'acquisitions NACQ (R,T) pour la période T définie par les bornes O et t ;
   - un volume "en cours d'utilisation" VUTI (R,t) représente le nombre de requêtes pour lesquelles la portion de ressource R a été acquise mais

pas encore libérée au temps t. Ce volume correspond donc à la différence entre les nombres d'acquisitions NACQ (R,T) et de libérations NLIB (R,T) pour la période T définie par les bornes O et t ;

- un volume "en cours de service" VSER (R,t) représente le nombre de requêtes pour lesquelles la portion de ressource R a été requise mais pas encore libérée au temps t. Ce volume correspond donc à la différence entre les nombres de réquisitions NREQ (R,T) et de libérations NLIB (R,T) pour la période T entre O et t. Ce volume est la somme des deux précédents ;
- la valeur moyenne de la distribution de probabilité de ces volumes peut être déterminée par :

$$\frac{1}{T} \int_0^T \text{volume } (R,t)\ dt$$

où volume (R,t) représente respectivement VREQ (R,t), VUTI (R,t) et VSER (R,t) ;

c.2) la **capacité disponible** de la ressource R à un moment donné dans la période T considérée, sachant que :

- la valeur moyenne de la distribution de probabilité correspondante peut être déterminée par :

$$\frac{1}{T} \int_0^T CD(R,t)\ dt$$

où CD (R,t) représente la capacité disponible de la ressource R au temps t.

Comme pour les autres résultats statistiques concernant des volumes, cette moyenne "temporelle" peut facilement être calculée car chaque changement d'état d'une ressource (réquisition, acquisition ou libération) a donné lieu, pendant la simulation, à l'enregistrement d'une mesure dans la base de données statistiques.

On peut imaginer d'autres résultats statistiques caractérisant le comportement d'une ressource. En particulier, on peut souhaiter obtenir des résultats statistiques sur la **capacité utilisée** d'une ressource R à un moment

donné, dont la valeur moyenne est déterminée par :

$$\frac{1}{T} \int_0^T CU(R,t)\ dt$$

où CU (R,t) représente la capacité utilisée de la ressource R au temps t.

On peut également dégager un **taux d'utilisation** moyen d'une ressource R sur une période T donnée, déterminé de la façon suivante :

$$\frac{\int_0^T CU\ (R,t)\ dt}{\int_0^T CMD\ (R,t)\ dt}$$

où CMD (R,t) est la capacité maximum disponible de la ressource R au temps t, qui est fixée par son calendrier de disponibilité. On notera que CMD (R,t) = CU (R,t) + CD (R,t).

Quelques précautions doivent cependant être prises dans le calcul de ce taux moyen d'utilisation pour tenir compte du calendrier de disponibilité de la ressource qui peut spécifier une capacité maximale disponible nulle pendant certaines périodes.

Enfin, on pourrait vouloir obtenir les résultats statistiques concernant les requêtes (a), b), c.1)) répartis par classe de processus requérants. Toutefois, ces résultats n'ajouteraient aucune information supplémentaire par rapport aux résultats sur l'évolution des processus. En effet, notre hypothèse d'allocation globale signifie que toutes les requêtes d'un processus suivent exactement la même évolution que le processus (cf. I.3.5.). Il suffit donc de se rapporter aux résultats statistiques des classes de processus requérant une portion de la ressource considérée pour avoir l'information souhaitée.

### II.4.3. Entrées-Sorties

Les **paramètres** de cet outil servent à déterminer le contenu et la forme des rapports (= sorties) à produire à partir des mesures

élémentaires enregistrées dans la base de données statistiques. Ils constituent l'unique entrée de l'outil d'exploitation. Ils permettent notamment de mentionner :

1. les **classes d'objets** pour lesquelles on veut obtenir des résultats statistiques. En fonction du type d'objet, l'outil est capable de détecter automatiquement quels sont les résultats statistiques qui caractérisent le comportement des objets sélectionnés ;
2. les **périodes de référence,** à l'intérieur de la période simulée, pour lesquelles on souhaite produire des résultats statistiques. Ceux-ci peuvent être présentés sous forme de valeur
   - "globales" pour une période de référence
     (exemple : du 5ème au 20ème jour de la simulation) ;
   - "périodiques" par période se répétant dans une période plus large
     (exemple : heure par heure d'une journée moyenne reconstruite à partir de la 5ème à la 20ème journée de la simulation) ;
   - "chronologiques" pour des périodes fines à l'intérieur d'une période de référence plus large
     (exemple : pour toutes les heures du 5ème au 20ème jour) ;
3. la **forme de présentation** souhaitée des résultats statistiques :
   tableaux de chiffre, courbes, diagrammes ...

- CHAPITRE III -

# VERIFICATION DES SPECIFICATIONS DU COMPORTEMENT DES SYSTEMES D'INFORMATION

## III.1. INTRODUCTION

Après avoir spécifié le comportement d'un système d'information à l'aide du modèle proposé au chapitre 1, encore faut-il **vérifier** qu'il s'agit d'une **bonne spécification** de la solution choisie en s'assurant de l'**utilisation correcte du modèle.**

La vérification d'un schéma de comportement comprend trois aspects qui correspondent aux qualités attendues des spécifications de comportement : **complétude, cohérence** et **faisabilité.** Il faut en effet pouvoir s'assurer qu'un schéma est :

complet en vérifiant qu'il n'existe pas de composants référencés mais non complètement décrits ;

cohérent en vérifiant qu'il n'existe pas de contradiction dans la description. La cohérence doit non seulement être vérifiée au sein d'un schéma (cohérence interne) mais aussi entre les différents niveaux de décomposition en sous-schémas (cohérence externe) ;

faisable en vérifiant que le schéma représente un comportement respectant les performances attendues du système à réaliser.

La vérification de ces 3 propriétés peut être partiellement automatisée et en tout cas **assistée par** l'utilisation des outils disponibles dans l'**environnement logiciel** généralisé adopté, complété par l'**outil d'évaluation** présenté au chapitre précédent.

Ce chapitre présente les contrôles à effectuer pour respectivement vérifier la complétude, la cohérence et la faisabilité des spécifications ; il précise, en outre, l'assistance fournie par l'utilisation des outils logiciels disponibles.

Il faut cependant noter qu'une **bonne spécification** de la solution choisie, résultant d'une utilisation correcte des modèles, ne garantit pas la spécification d'une **bonne solution** c'est-à-dire :

- conforme à l'attente et aux besoins de l'organisation ;
- adaptée aux possibilités technologiques, humaines, financières et organisationnelles de l'organisation.

Contrairement à la vérification des trois qualités mentionnées ci-dessus, cette validation, essentiellement qualitative, des spécifications ne peut guère être assistée par l'utilisation d'outils logiciels et ne sera donc pas développée dans ce chapitre.

## III.2. COMPLETUDE

### III.2.1. Présentation générale

Un schéma de comportement est **COMPLET** si chaque composant de la spécification est complètement décrit, c'est-à-dire s'il n'existe plus de composants qui aient été référencés mais dont la description est incomplète et reste à déterminer.

La vérification de la complétude revient donc à s'assurer que chaque classe d'objets reprise dans la spécification possède bien, en fonction de son type, toutes les caractéristiques prévues dans le modèle retenu et exposé au chapitre 1. Il ne s'agit donc pas à ce niveau de vérifier si des classes d'objets et/ou des liaisons entre elles manquent dans la spécification.

**Outil logiciel de sélection**

Dans le contexte de l'environnement logiciel adopté, ces contrôles de complétude peuvent être effectués à l'aide d'un **langage d'interrogation** de haut niveau intégré dans cet environnement (cf. Préambule), dont la fonction est de **sélectionner** dans la base de données de spécification **des composants** qui **satisfont** certains **critères.**

Il s'agit d'un langage ensembliste : l'exécution d'une question crée un ensemble de noms de classes obtenus par l'intersection (AND) et/ou l'union (OR) d'ensembles définis par des critères de sélection. Les critères de sélection sont des expressions booléennes dont les opérandes sont des instructions du langage de spécification dans lesquelles on a substitué certaines parties par :

- ? pour mentionner l'objet de la recherche ;
- ! pour signaler l'existence d'au moins un composant d'un type donné.

La question suivante permet par exemple de retrouver dans la base de données toutes les classes de processus qui (?) causent la survenance d'au moins un (!) événement :

PROCESSUS AND ? CAUSE !

Le fait de pouvoir **sélectionner,** à l'aide de ce langage, des **composants qui ne possèdent pas** - encore - certaines caractéristiques et donc ne satisfont pas certains critères de sélection permet d'envisager l'utilisation de cet outil pour détecter des spécifications incomplètes.

Cette section présente les **contrôles de complétude** à effectuer pour garantir que **chaque** composant de la spécification d'un comportement est **complètement** décrit. A chaque contrôle mentionné est également associé le **critère de sélection** du langage d'interrogation qui permet de détecter automatiquement les classes d'objets dont la spécification reste à compléter.

### III.2.2. Règles de complétude

Un schéma de comportement est **complètement** spécifié si :

1. pour toutes les classes de **processus** citées dans ce schéma :

   a) leur procédure est décrite - sous forme d'un texte descriptif en format libre :

   PROCESSUS AND NOT (HAS PROCEDURE)

   b) leur durée estimée d'exécution est précisée :

   PROCESSUS AND NOT (? ACTIF PENDANT !)

   c) leur nom caractérise au moins une classe de survenances pour garantir que, parmi les transformations d'informations véhiculées et/ou enregistrées par les processus d'une classe, certaines ont une incidence sur le comportement représenté par le schéma :

   PROCESSUS AND NOT (? CAUSE !)

   d) la probabilité d'être vraie est fixée pour chaque condition de survenance référencée :

CONDITION AND NOT (? VRAIE DANS !)

e) leur nom caractérise au moins une classe de déclenchements pour garantir que le déclenchement de leurs processus a été prévu (PRE-CONDITION) :

PROCESSUS AND NOT (? DECLENCHE PAR !)

f) le prédicat de chaque condition de déclenchement citée est défini :

CONDITION AND NOT (HAS PREDICAT)

2. pour toutes les classes d'**événements** citées dans le schéma :

a) les attributs d'événements, sur lesquels porte une modalité de déclenchement, sont spécifiés :

! ATTRIBUT EST ? AND NOT
(? VALEUR ! OR ? DOMAINE DE VALEUR ! OR ? DISTRIBUTION !)

b) leur nom caractérise au moins une classe de survenances s'il s'agit d'événements internes ou leur échéancier a été défini s'il s'agit d'événements (externes) initiaux :

EVENEMENT AND NOT
(? CAUSE PAR ! OR ? SURVIENT ! FOIS PENDANT !)

c) leur nom caractérise au moins une classe de déclenchements s'il ne s'agit pas d'événements terminaux :

EVENEMENT AND NOT (? DECLENCHE !)

Les seules classes d'événements vérifiant ce critère doivent alors correspondre aux classes d'événements terminaux du schéma analysé ;

3. pour toutes les **ressources** référencées - citées - dans le schéma :

a) leur calendrier de disponibilité est décrit :

```
RESSOURCE AND NOT (? CAPACITE ! DISPONIBLE PENDANT !)
```

b) toutes les périodes reprises dans la spécification d'un calendrier de disponibilité sont précisées :

```
(? CAPACITE ! DISPONIBLE PENDANT ?)
    AND NOT (? DE ! A ! DANS !)
```

c) leur nom caractérise au moins une classe de réquisitions :

```
RESSOURCE AND NOT
(? REQUISE PAR ! OR ? PARTAGEE PAR !)
```

d) tous les taux de réquisition, non exprimés sous forme d'une constante, sont fixés par une distribution de probabilité :

```
(! REQUISE PAR ! AU TAUX DE ?)
    AND NOT (? DISTRIBUTION !)
```

## III.3. COHERENCE

### III.3.1. Présentation générale

Un schéma de comportement est **cohérent** si :

1. (cohérence interne) les spécifications du schéma ne sont pas contradictoires entre elles et garantissent que tous les traitements décrits par les classes de processus sont exécutables ou, de façon duale, que tous les états représentés par les classes d'événements sont **accessibles** ;
2. (cohérence externe) les spécifications du schéma ne sont pas contradictoires avec celles du schéma plus **agrégé** dont il constitue un sous-schéma.

Contrairement aux contrôles de complétude (cf. III.2.), aucun outil standard de l'environnement logiciel adopté ne permet d'effectuer automatiquement - sans intervention du concepteur - les contrôles de cohérence présentés dans cette section. La vérification de la cohérence s'apparente en effet aux **preuves de programmes** dans lesquelles on tâche de vérifier la correction totale (terminaison) et partielle (conformité aux spécifications si terminaison). Cette **approche par démonstration** ne sera pas

développée dans cette étude même si elle doit être considérée comme une extension normale de celle-ci.

Par contre, on peut imaginer deux approches moins ambitieuses et plus rudimentaires pour vérifier la cohérence des spécifications dans lesquelles l'utilisation des outils disponibles (cf. Préambule) peut constituer une forme d'assistance appréciable.

La première approche, dite **par visualisation,** consiste à présenter, à l'aide des outils logiciels standards, les spécifications du comportement analysé sous une forme plus propice à l'analyse - par le concepteur. - que la forme sous laquelle elles ont été initialement rédigées et enregistrées dans la base de données.

La seconde approche, dite **par exécution,** consiste à utiliser l'outil d'évaluation décrit au chapitre précédent, et simuler - donc exécuter - le comportement spécifié et tâcher de détecter des incohérences sur base de la "trace" fournie à l'exécution.

Cette section précise pour les deux aspects de la cohérence mentionnés ci-dessus

1. les contrôles à effectuer ;
2. l'utilisation qui peut être faite
   a) des outils standards, en particulier d'un outil de structuration brièvement présenté ci-dessous, disponibles dans l'environnement logiciel de spécification,
   b) de l'outil d'évaluation par simulation ;

pour détecter des incohérences dans les spécifications d'un comportement.

**Outil logiciel de structuration**

L'outil privilégié pour restituer les spécifications est l'**outil de structuration** disponible dans l'environnement logiciel généralisé adopté (cf. Préambule). Cet outil implémente en fait un algorithme de **fermeture transitive** d'un graphe qui :

1. sélectionne dans la base de données des spécifications les classes d'objets d'un type donné qui succèdent à - précèdent - une classe d'objets donnée via des associations d'un type donné ;

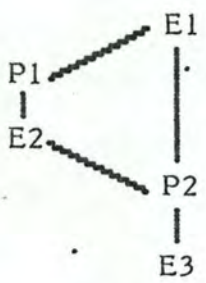
2. présente, par niveau et sous forme d'une liste "indentée", d'une matrice ou d'un graphe, les classes d'objets successeurs - prédécesseurs -
où les classes d'objets d'un niveau i sont les successeurs - prédécesseurs - immédiats des classes d'objets d'un niveau i-1 et les classes du niveau 1 sont les classes pour lesquelles le rapport est demandé (= paramètres de l'outil).

Cet outil de structuration permet notamment de reconstituer automatiquement un schéma de comportement, préalablement spécifié et enregistré dans la base de données, au départ des seules classes d'événements initiaux ou terminaux d'un schéma.
Dans ce cas, les seules classes prises en considération sont les classes de **processus** et **d'événements** qui, via les associations de **déclenchements** et de **survenances,** succèdent à - précèdent - la ou les classes d'événements initiaux - terminaux - pour lesquelles on demande le rapport.

Exemple III.1

Supposons que les spécifications suivantes aient été enregistrées dans la base de données :

```
DEF PROCESSUS P1
    DECLENCHE PAR E1
    CAUSE E2

DEF PROCESSUS P2
    DECLENCHE PAR E1, E2
    CAUSE E3
```




L'outil de structuration permet, par exemple, au départ de la seule classe d'événements terminaux E3, de restituer ces spécifications de la façon suivante :

```
1 E3          EVT
  2 P2        PRC      cause E3
    3 E1      EVT      déclenche P2
    3 E2      EVT      déclenche P2
      4 P1    PRC      cause E2
        5 E1  EVT  *   déclenche P1
```

On notera la présence de l'* qui signale que cette classe (E1) est déjà apparue dans la structure et intervient donc dans plusieurs associations.

Ce rapport met donc clairement en évidence les "filières" possibles ou suite de prédécesseurs (successeurs) entre deux classes d'événements.

### III.3.2. COHERENCE INTERNE : Règle d'accessibilité

La cohérence **interne** d'un schéma ne fait référence à aucune autre spécification externe au schéma et assure uniquement un comportement **sans blocage intempestif.** Cela signifie que les spécifications n'empêchent pas systématiquement le déclenchement de certains processus ou la survenance de certains événements.

Le contrôle de la cohérence interne peut être comparé à la vérification de l'atteignabilité dans les Réseaux de Petri [BRAMS, 1983]. Il a de plus été montré [HECKENROTH, 1979] qu'il était possible :

- de transformer des spécifications, faites à l'aide d'un modèle proche de celui présenté au chapitre 1, en un Réseau de Petri ;
- d'utiliser les outils classiques d'analyse des Réseaux de Petri et notamment de vérifier l'atteignabilité ("vivance").

Même si cette solution semble prometteuse et constituer un moyen privilégié pour vérifier la cohérence interne, elle n'a cependant pas été retenue dans cette étude étant donné :

- la complexité de la transformation automatique dans un autre formalisme et les problèmes posés par l'interprétation des résultats dans un formalisme différent de celui adopté initialement ;
- l'importance attachée au temps (notion de délai) dans le modèle adopté qui complique les algorithmes d'analyse ;
- la préférence accordée dans cette étude à l'évaluation par simulation - sans transformation de modèle.

**Règle d'accessibilité**

La cohérence interne est vérifiée si la règle d'accessibilité suivante est respectée :

> pour les classes d'événements $E_i$ (de processus $P_i$) d'un schéma, en particulier les classes d'événements terminaux,
> il existe une collection d'événements initiaux telle qu'un événement $E_i$ (un processus $P_i$) soit successeur de chaque événement de cette collection.

La mise en oeuvre de cette règle suppose qu'étant donné la structure d'un schéma :

A. les modalités de déclenchement et de survenance ne sont pas contradictoires entre elles ;

B. les propriétés des requêtes ne sont pas contradictoires avec le calendrier de disponibilité des ressources requises ;

C. les échéanciers de classes d'événements initiaux (externes) ne sont pas contradictoires avec les modalités de déclenchement du schéma.

A. **Les modalités de déclenchement et de survenance sont contradictoires entre elles** lorsqu'une situation de décision empêche systématiquement une situation de synchronisation de se produire car certains événements contribuant au déclenchement d'un même processus ne peuvent jamais être simultanément présents.

Les trois schémas exposés ci-dessous illustrent les cas les plus représentatifs de situations d'incohérence interne dans les modalités de déclenchement et de survenance. Elles se distinguent par les propriétés d'événements sur lesquelles portent certaines contraintes de déclenchement :

a) attributs d'événements ;
b) délais entre la survenance d'événements et leur contribution au déclenchement d'un processus ;
c) nombre de déclenchements de processus auxquels un événement peut contribuer.

Les trois cas suivants (a, b et c) supposent que l'attribut A2 identifie les événements successeurs d'un même événement initial (E1).

Cas a)

Soit un schéma spécifié de la façon suivante

```
DEF PROCESSUS P1
    DECLENCHE PAR E1 QUAND C1
    CAUSE E2

DEF PROCESSUS P2
    DECLENCHE PAR E1, E2 QUAND C2
    CAUSE E3

DEF CONDITION C1
    PREDICAT : A1 DE E1 = V1

DEF CONDITION C2
    PREDICAT : A1 DE E1 ≠ V1 ET
               A2 DE E1 = A2 DE E2
```

Ce schéma est incohérent car :

- un même événement E1 contribue au déclenchement d'un processus P1 ou d'un processus P2 (DECISION) alors que
- le déclenchement d'un processus P2 suppose la survenance d'un événement E1 et d'un événement E2 (SYNCHRONISATION), lui-même successeur de l'événement E1.

Cas b)

Soit un schéma spécifié de la façon suivante :

```
DEF PROCESSUS P1
    DECLENCHE PAR E1
    CAUSE E2

DEF PROCESSUS P2
    DECLENCHE PAR E1, E2 QUAND C1
    ACTIF PENDANT "10 min"
    CAUSE E3

DEF CONDITION C1
    PREDICAT :
       DELAI ENTRE SURVENANCE DE E1 ET DECLENCHEMENT DE
       P2 = 0 ET A1 de E1 = A1 de E2
```

Ce schéma est incohérent car :

- un événement E2 devrait survenir en même temps que son prédecesseur E1 pour déclencher un processus P2 alors que
- la durée du processus P1 sépare les deux survenances.

Cas c)

Soit un schéma spécifié de la façon suivante :

```
DEF PROCESSUS P1
    CAUSE 1 E1 SI C1
          1 E2 SI PAS-C1

DEF PROCESSUS P2
    DECLENCHE PAR E1, E2 QUAND C3

DEF CONDITION C3
    PREDICAT : NOMBRE DE E1 DECLENCHANT LE MEME
               P2 = 1 ET
               NOMBRE DE E2 DECLENCHANT LE MEME
               P2 = 1 ET
               A1 DE E1 = A1 DE E2
```

Ce schéma est incohérent car :

- un processus P1 cause la survenance d'un événement E1 **OU** d'un événement E2 (DECISION) alors que
- le déclenchement d'un processus P2 suppose la survenance d'un événement E1 **ET** d'un événement E2 (SYNCHRONISATION).

B. **Les propriétés des requêtes sont contradictoires avec le calendrier de disponibilité des ressources requises** lorsque :

a) un processus est non interruptible et que sa durée estimée d'exécution, c'est-à-dire la durée de ses réquisitions, est supérieure à la période la plus large du calendrier de disponibilité des ressources requises ;

b) la durée d'exécution d'un processus non interruptible est supérieure à l'intersection des périodes les plus larges du calendrier de disponibilité des ressources requises par le processus ;

c) un taux de réquisition est supérieur à la capacité maximum disponible de la ressource requise.

Les cas suivants illustrent ces trois situations d'incohérence interne :

a) un processus non interruptible P1 dure 2 heures et requiert une ressource R1 qui est disponible entre 8 et 9 heures chaque jour ;

b) un processus non interruptible P1 requiert une ressource R1 disponible entre 8 et 12 heures et requiert une ressource R2 disponible entre 14 et 16 heures ;

c) un processus P1 requiert 10 unités d'une ressource R1 qui a une capacité maximale disponible de 5 unités.

C. **L'échéancier d'une classe d'événements initiaux est contradictoire avec les modalités de déclenchement d'un schéma** lorsque le nombre d'événements qui contribuent au déclenchement d'un processus P1 n'est jamais suffisant.

Le schéma suivant est incohérent car un processus P1 ne pourra jamais être déclenché :

- 100 événements E1 contribuent au déclenchement d'un processus P1 à condition que le délai entre leur survenance et leur contribution n'excède pas 1 journée ;
- mais l'échéancier de la classe d'événements E1 prévoie seulement la survenance de 10 événements par jour.

Exemple III.2.

Le schéma dont les spécifications suivent décrit un comportement fonctionnel cohérent, car ses modalités de déclenchement et de survenance ne sont pas contradictoires :

DEF PROCESSUS enregistrement-d'une-commande
DECLENCHE PAR commande-du-jour
CAUSE 1 commande-à-enregistrer SI commande-valide
1 commande-rejetée SI commande-invalide

DEF PROCESSUS enregistrement-d'une-réquisition
DECLENCHE PAR commande-à-enregistrer, commande-différée
CAUSE 1 réquisition-à-préparer SI stock-suffisant
1 commande-à-différer SI stock-insuffisant

DEF PROCESSUS préparation-réquisition
DECLENCHE PAR réquisition-à-préparer
CAUSE 1 commande-préparée

La règle d'accessibilité est en effet vérifiée pour les trois classes d'événements terminaux :

a) la survenance d'une "commande rejetée" est possible au départ de la survenance d'une "commande du jour" puisqu'un processus "enregistrement d'une commande" peut causer la survenance d'une "commande rejetée" (SI "commande invalide") ;
b) la survenance d'une "commande à différer" peut succéder à la survenance d'une "commande du jour" OU d'une "commande différée" puisqu'un processus "enregistrement d'une réquisition" peut causer la survenance d'une "commande à différer" (SI "stock insuffisant") ;
c) enfin, la survenance d'une "commande préparée" peut succéder à la survenance d'une "commande du jour" OU d'une "commande différée" (SI "commande valide" ET "stock suffisant").

Dans le contexte de l'environnement logiciel adopté, l'outil de génération intégré dans l'outil d'évaluation par simulation peut détecter automatiquement les contradictions (B.) entre les requêtes et les calendriers de disponibilité des ressources requises puisque ces incohérences empêchent le déroulement d'une simulation.

Par contre et dans l'état actuel des développements, seule une approche - non automatique - par visualisation ou par exécution, permet de mettre en évidence les contradictions entre les modalités de déclenchement, celles de survenance (A.) et les échéanciers (C.) d'un schéma.

### III.3.2.1. Contrôle de la cohérence interne par visualisation

Les outils standards de l'environnement logiciel adopté peuvent être utilisés pour faciliter la détection des contradictions éventuelles entre les

échéanciers des classes d'événements initiaux, les modalités de déclenchement et celles de survenance d'un schéma (A. et B.).

La démarche proposée est la suivante :

1. Sélection des classes d'événements terminaux du schéma à l'aide du langage d'interrogation, déjà présenté dans le contexte du contrôle de la complétude (III.1) :

   EVENEMENT AND ? CAUSE PAR !
   AND NOT (? DECLENCHE !)

   Après s'être assuré que les classes d'événements sélectionnées correspondent aux classes d'événements terminaux dont il s'agit de vérifier l'accessibilité, les deux étapes suivantes doivent alors être répétées pour chaque classe d'événements $ET_i$ sélectionnée ;

2. Production automatique d'un rapport de structuration restituant la partie du schéma comprenant les classes d'événements et de processus **prédécesseurs** des événements de la classe $ET_i$ considérée ;

3. Analyse du rapport produit pour s'assurer que :

   a) les dernières classes d'événements prédécesseurs découvertes dans le rapport correspondent bien à des classes d'événements initiaux du schéma ;

   b) les modalités de survenance et de déclenchement ainsi que les échéanciers ne sont pas contradictoires.

Exemple III.3.

Dans le cas des spécifications correspondant au **cas a)** présenté ci-dessus (cf. p. 99), la démarche serait appliquée de la façon suivante :

1. sélection des classes d'événements terminaux :

   E3 est bien la seule classe dont les événements sont causés par mais ne déclenchent aucun processus ;

2. production automatique du rapport de structuration suivant pour retrouver les classes d'événements initiaux prédécesseurs des événements E3 :

```
1 E3          EVT
 2 P2         PRC   cause  E3
   3 E1       EVT   déclenche P2 quand C2
   3 E2       EVT   déclenche P2 quand C2
     4 P1     PRC   cause E2
       5 E1   EVT * déclenche P1 quand C1
```

C1 == A1 DE E1 = V1
C2 == A1 DE E1 ≠ V1 ET A2 DE E1 = A2 DE E2

3. l'analyse de ce rapport devrait mettre en évidence que :

   a) E1, qui n'a aucun prédécesseur dans ce rapport, est bien la seule classe d'événements initiaux du schéma ; le fait qu'il apparaisse 2 fois dans le rapport signale la présence de 2 filières possibles pour atteindre un événement E3 au départ d'un événement E1 ;
   b) les conditions de déclenchement C1 et C2 sont contradictoires puisque pour accéder au même événement E3, l'attribut A1 devrait être à la fois :
      - (C1) égal à et
      - (C2) différent de la valeur V1.

Lorsque le schéma devient important et que le nombre de conditions de survenance et de déclenchement est assez élevé, la vérification de la cohérence interne par la méthode expliquée ci-dessus risque de s'avérer assez fastidieuse et peu sûre quant au résultat. La méthode suivante peut alors se justifier malgré son apparente lourdeur.

### III.3.2.2. Contrôle de la cohérence interne par exécution

L'outil d'évaluation par simulation, présenté au chapitre précédent, peut en effet servir à vérifier l'absence de contradictions dans un schéma de comportement, c'est-à-dire le non respect de la règle d'accessibilité mentionnée ci-dessus.

La démarche proposée consiste à répéter la procédure suivante pour chaque classe d'événements terminaux $ET_i$, sélectionnée de la même façon que dans l'approche par visualisation, c'est-à-dire en ayant recours au langage d'interrogation.

1. Sur base d'un rapport de structuration équivalent à celui utilisé dans le contrôle par visualisation :

   a) fixation d'un échéancier simplifié qui comporte une seule occurrence pour chaque classe d'événements initiaux ;
   b) modification des spécifications définies pour forcer :
      - la probabilité d'être vraie des conditions de survenance à 100 % ;
      - la valeur des attributs intervenant dans une condition de déclenchement de telle sorte qu'elle soit vraie.

   Ces deux dernières actions doivent être effectuées pour obtenir un comportement déterministe de la simulation et garantir que le

comportement simulé correspondra à la partie du schéma qui mène aux événements terminaux considérés ;

2. Simulation du comportement correspondant à la partie du schéma restituée par le rapport de structuration produit à l'étape précédente, **sans prendre en considération les réquisitions de ressources ;**

3. Analyse de la "trace" produite par la simulation pour s'assurer qu'au moins un événement terminal de la classe $ET_i$ considérée est survenu en cours de simulation.

Exemple III.4.

Dans le cas des spécifications correspondant au **cas b)** présenté ci-dessus (cf. p. 100) la démarche serait appliquée de la façon suivante pour vérifier l'accessibilité aux événements - terminaux - de la classe E3 :

1. Production automatique du rapport de structuration suivant pour retrouver les classes d'événements initiaux prédécesseurs des événements E3 :

```
1 E3            EVT
  2 P2          PRC     cause E3
    3 E1        EVT     déclenche P2 quand C1
    3 E2        EVT     déclenche P2 quand C1
      4 P1      PRC     cause E2
        5 E1    EVT  *  déclenche P1
```

C1 == délai entre survenance de E1 et déclenchement de P2 = 0 ET A1 de E1 = A1 de E2

1.a. fixation d'un échéancier simplifié pour la classe des événements initiaux E1 :

```
DEF EVENEMENT E1
    SURVIENT 1 FOIS
```

1.b. modification des spécifications pour garantir un comportement simulé conforme au rapport de structuration précédent :

```
DEF ATTRIBUT A1
    VALEUR 1
```

2. Lancement d'une simulation pour laquelle l'outil de génération intégré aurait automatiquement sélectionné, au départ de la classe des événements initiaux E1, la partie de schéma qui correspond au rapport de structuration précédent ;

3. La "trace" suivante qui aurait été produite en cours de simulation serait ensuite analysée :

```
0:00:00     survenance de E1(1) [initial]
0:00:00     déclenchement de P1(1)
0:00:00     *** condition C1 pour déclenchement de P2
            non vérifiée
0:00:00     démarrage de P1(1)
```

0:10:00 terminaison de P1(1)
0:10:00 survenance de E2(1)
0:10:00 *** condition C1 pour déclenchement de P2 non vérifiée
0:10:00 FIN DE SIMULATION

La simple lecture de cette "trace" permettrait de constater qu'aucun processus P2 n'a pu être déclenché car la condition C1 n'a jamais été vérifiée et donc qu'aucun événement E3 n'est survenu pendant la simulation.

### III.3.3. COHERENCE EXTERNE : règle de (dés-)agrégabilité

La cohérence externe d'un schéma fait référence à des spécifications extérieures au schéma. La cohérence d'un schéma ne peut, en effet, être complètement vérifiée que par rapport à des spécifications externes et ne peut donc être envisagée que dans un **contexte de décomposition.** La validation d'un schéma revient alors à vérifier la **cohérence entre 2 niveaux consécutifs** de décomposition. Cela signifie que les spécifications externes d'un sous-schéma sont celles données au niveau - supérieur - de la classe de processus dont le sous-schéma constitue une expression plus affinée du comportement.

La vérification de la cohérence externe peut être apparentée au contrôle du "balancing" dans les approches "data flow" [DE MARCO, 1979]. Le "balancing" correspond à "la relation qui existe entre un diagramme-parent et ses diagrammes-enfants. Plus spécifiquement, il représente l'équivalence des entrées/sorties d'un traitement du diagramme-parent et les entrées origines/ sorties destinations du diagramme-enfant correspondant". Toutefois, contrairement au contrôle de la cohérence externe, le contrôle du "balancing" porte essentiellement sur le nom des entrées-sorties et n'intègre pas les composantes dynamiques temporelles et de synchronisation.

**Règle de (dés-)agrégabilité**

Pour garantir qu'un schéma S modélise le comportement équivalent à celui des processus d'une classe P d'un schéma plus agrégé (= du niveau de décomposition supérieur), la **règle de (dés-)agrégabilité** suivante doit être respectée :

- d'une part, les classes d'événements $E_i$ contribuant au déclenchement des processus P sont les classes d'événements initiaux du (sous-) schéma S. De plus, ce dernier ne contredit pas les modalités de contribution des événements $E_i$ au déclenchement des processus P ;

- d'autre part, les classes d'événements $E_j$ dont la survenance est causée par les processus P sont les classes d'événements terminaux du (sous-) schéma S. Et ce dernier respecte également les modalités de survenance des événements $E_j$ causée par les processus P.

La mise en oeuvre de cette règle suppose qu'étant donné un (sous-) schéma S modélisant le comportement des processus P :

a) si les seuls événements, qui auraient entraîné le déclenchement d'un processus P dans le schéma plus agrégé, survenaient (= vérification de la Pré-condition),
alors seuls les événements, dont la survenance aurait été causée par un processus P, surviendraient après un certain temps. (= vérification de la Post-condition) ;

b) les échéanciers des classes d'événements initiaux du schéma S reproduisent - éventuellement à des dates différentes - les mêmes survenances d'événements qui auraient contribué au déclenchement des processus P dans le schéma plus agrégé.

Par contre, la règle de (dés-)agrégabilité ne s'applique pas aux contraintes de moyens car le degré de finesse apporté à la description des ressources et de leurs réquisitions par les processus dépend précisément du niveau de décomposition des traitements auquel on se situe.
Les deux exemples suivants illustrent respectivement une situation où la cohérence externe n'est pas vérifiée et où elle l'est.

Exemple III.5.

Soit une classe de processus P spécifiée de la façon suivante :

```
DEF PROCESSUS P
    DECLENCHE PAR E1, E2 QUAND C1
    CAUSE 1 E3
```

où la condition de déclenchement C1 précise que :

```
DEF CONDITION C1
    PREDICAT :
        1 E1 ET 1 E2
```

Soit un (sous-) schéma S censé représenter le comportement des processus P :

```
DEF PROCESSUS P1
    DECLENCHE PAR E1
    CAUSE 1 E3

DEF PROCESSUS P2
    DECLENCHE PAR E2
    CAUSE 1 E3
```

Le schéma S est contradictoire avec les spécifications de la classe de processus P car la survenance d'un événement E1 et d'un événement E2 (= précondition de P) entraîne à terme la survenance de 2 événements E3 alors qu'un processus P ne cause la survenance que d'un seul événement E3.

Exemple III.6.

Le sous-schéma suivant est un sous-schéma cohérent du schéma correspondant à l'exemple III.2. (cf. p. 102) puisque les spécifications de la classe de processus "Préparation réquisition" sont respectées par ce sous-schéma :

```
DEF PROCESSUS      ordonnancement-série
    DECLENCHE PAR réquisition-à-préparer QUAND série-
                  constituée
    CAUSE 1 série-à-préparer

DEF CONDITION      série-constituée
    PREDICAT :
        NOMBRE DE réquisition-à-préparer
        DECLENCHANT LE MEME ordonnancement-série = 10

DEF PROCESSUS      préparation-série
    DECLENCHE PAR série-à-préparer
    CAUSE 10 commande-préparée
```

En effet,

a) dans le schéma de l'exemple III.2.,
un processus "préparation réquisition" est déclenché par la survenance d'1 événement "réquisition à préparer" et cause la survenance d'1 événement "commande préparée" ;

b) dans ce schéma-ci,
- une "réquisition à préparer" est un événement initial du schéma ;
- une "commande préparée" est un événement terminal du schéma ;
- une "réquisition à préparer" précède toujours 1 "commande préparée" puisque la survenance de 10 "réquisitions à préparer" cause la survenance de 10 "commandes préparées".

Comme dans le cas de la cohérence interne, une approche - non automatique mais assez systématique - par visualisation ou par simulation est susceptible d'apporter une assistance au concepteur dans la direction des incohérences externes d'un schéma de comportement.

### III.3.3.1. Contrôle de la cohérence externe par visualisation

Les outils standards de l'environnement logiciel généralisé adopté peuvent apporter une certaine aide dans le contrôle de la cohérence externe des spécifications.

En particulier, le langage d'interrogation, présenté dans le cadre des contrôles de complétude, peut être utilisé pour s'assurer que :

a) les classes d'événements contribuant au déclenchement des processus de la classe P sont les classes d'événements initiaux du sous-schéma S qui spécifie le comportement des processus P ;
b) les classes d'événements dont la survenance est causée par les processus P sont les classes d'événements terminaux du sous-schéma S.

Ces deux contrôles peuvent être automatiquement effectués à l'aide des questions suivantes :

1. sélection des classes d'événements contribuant au déclenchement de processus P :

   evt-décl-P = EVENEMENT AND ? DECLENCHE P

2. sélection des classes d'événements dont la survenance est causée par des processus P :

   evt-causé-par-P = EVENEMENT AND ? CAUSE PAR P

3. sélection des classes de processus du sous-schéma S, c'est-à-dire de sous-processus de processus P :

   ss-proc-P = PROCESSUS AND ? SOUS-PROCESSUS DE P

4. sélection des classes d'événements initiaux du sous-schéma S :

   evt-init-S = EVENEMENT AND ? DECLENCHE ss-proc-P
   AND NOT (? CAUSE PAR ss-proc-P)

5. sélection des classes d'événements terminaux du sous-schéma S :

   evt-term-S = EVENEMENT AND ? CAUSE PAR ss-proc-P
   AND NOT (? DECLENCHE ss-proc-P)

6. détection des classes d'événements contribuant au déclenchement de processus P mais non initiaux du schéma S :

   evt-decl-P AND NOT evt-init-S

7. détection des classes d'événements initiaux du schéma S mais ne contribuant pas au déclenchement de processus P :

   evt-init-S AND NOT evt-décl-P

8. détection des classes d'événements dont la survenance est causée par les processus P mais non terminaux du schéma S :

   evt-causé-par-P AND NOT evt-term-S

9. détection des classes d'événements terminaux du schéma S mais dont la survenance n'est pas causée par les processus P :

   evt-term-S AND NOT evt-causé-par-P.

Toutefois, cette vérification à l'aide du langage d'interrogation ne porte que sur le nom des classes d'événements et ne permet donc pas de prendre en considération les modalités de déclenchement et de survenance traduisant les pré- et post- conditions dynamiques des processus de la classe P.

Il faut pour cela procéder à l'examen détaillé d'un rapport de structuration restituant le schéma. Après avoir vérifié la correspondance des noms à l'aide de la procédure d'interrogation mentionnée ci-dessus, la démarche pourrait être la suivante :

1. Production d'un rapport de structuration restituant toutes les filières au départ des classes d'événements contribuant au déclenchement de processus P - donc initiaux du schéma S ; la vérification préalable de la cohérence interne du schéma S garantissant que chaque filière aboutit nécessairement à une classe d'événements terminaux du schéma S ;
2. Examen des modalités de déclenchement et de survenance associées à la structure produite et détection - par visualisation - des éventuelles incohérences.

Exemple III.7.

Le rapport de structuration correspondant à la spécification de l'exemple III.5. (cf. p. 107), censé représenter le comportement des processus P, serait la suivante :

```
1 E1        EVT
  2 P1      PRC       déclenché par E1
    3 E3    EVT       causée par P1
1 E2        EVT
  2 P2      PRC       déclenché par E2
    3 E3    EVT   *   causé par P2
```

L'examen de ce rapport mettrait assez clairement en évidence que 2 événements E3 succèderaient systématiquement (pas de condition) à toute survenance d'1 événement E1 et d'1 événement E2, alors que la postcondition dynamique des processus P ne spécifiait la survenance que d'1 seul événement E3.

Signalons enfin un cas particulier mais typique d'incohérence. La vérification de la cohérence externe selon cette procédure peut en effet mettre en évidence la présence d'un **circuit** dans le schéma. Un circuit est détecté lorsqu'une classe d'événements se succède à elle-même dans le schéma, donc dans le rapport de structuration correspondant.

Il s'agit dans ce cas de s'assurer qu'il existe au moins une condition d'arrêt - sous forme d'une condition de survenance ou de déclenchement - dans la partie du schéma comprise entre les deux lignes du rapport où apparaît le nom de la classe d'événements repétée.

Exemple III.8.

Dans le cas de la spécification - partielle - suivante

```
DEF PROCESSUS   P1
    DECLENCHE PAR E1

DEF PROCESSUS P2
    DECLENCHE PAR E1
    CAUSE E2

DEF PROCESSUS P3
    DECLENCHE PAR E2
    CAUSE E1
```

La démarche mettrait en évidence un circuit infini par simple lecture du rapport de structuration suivant :

```
1 E1          EVT
  2 P1        PRC       déclenché par E1
  2 P2        PRC       déclenché par E1
    3 E2      EVT       causé par P2
      4 P3    PRC       déclenché par E2
        5 E1  EVT   *   causé par P3
```

*** pas de condition dans le circuit E1 - P2 - E2 - P3 - E1.

La présence d'un circuit infini traduit en effet une incohérence externe puisqu'elle signifie qu'à 1 événement initial risque de correspondre une infinité d'événements terminaux ; ce qui n'aura certainement pas été prévu dans la spécification de la classe de processus décomposés.

### III.3.3.2. Contrôle de la cohérence externe par exécution

L'outil d'évaluation par simulation, présenté au chapitre précédent peut également servir à vérifier l'absence d'incohérences externes entre un

schéma S et la classe de processus P dont il décrit le comportement.
La démarche proposée est la suivante :

1. Traduction - manuelle - sous forme d'échéancier pour toutes les classes d'événements initiaux du schéma S, de la précondition dynamique des processus P ;
2. Simulation du comportement décrit par le sous-schéma S, sans prendre en considération les réquisitions de ressources ;
3. Analyse de la trace produite par la simulation pour s'assurer que les événements terminaux dont la survenance a été causée en cours de simulation constituent bien une combinaison d'événements respectant la postcondition dynamique des processus P.

Exemple III.9.

Dans le cas des spécifications de l'exemple III.5., la démarche serait appliquée de la façon suivante :

1. la précondition des processus P (1 E1 et 1 E2) devrait être traduite par la spécification des 2 échéanciers suivants :

```
DEF EVENEMENT E1, E2
    SURVIENT 1 FOIS
```

2. une simulation serait exécutée pour le schéma S sachant que l'outil de génération aurait automatiquement sélectionné les classes correspondant au schéma S (E1, E2, P1, P2 et E3 ainsi que les classes de survenance et de déclenchement qui les relient) ;
3. la "trace" suivante qui aurait été produite en cours de simulation serait analysée :

```
0:00:00 survenance de      E1(1) [initial]
0:00:00 survenance de      E2(1) [initial]
0:00:00 déclenchement de P1(1) déclenché par E1(1)
0:00:00 déclenchement de P2(1) déclenché par E2(1)
0:00:00 démarrage de       P1(1)
0:00:00 démarrage de       P2(1)
0:00:00 terminaison de     P1(1)
0:00:00 terminaison de     P2(1)
0:00:00 survenance de      E3(1) causée par P1(1) [terminal]
0:00:00 survenance de      E3(2) causée par P2(1) [terminal]
0:00:00 FIN DE LA SIMULATION
```

La simple lecture des 2 dernières lignes de cette trace mettrait clairement en évidence le non respect de la postcondition des processus P (1 E3) puisque la survenance de 2 E3 a été causée.

## III.4. FAISABILITE

### III.4.1. Présentation générale

Un schéma de comportement est **FAISABLE** si le **comportement** modélisé, compte tenu du niveau d'utilisation des **ressources** et de la **charge** estimée du système, garantit le respect des **performances attendues** de la solution retenue.

La vérification de ce critère suppose donc que :

1. les **performances attendues** aient été spécifiées. Ces performances sont essentiellement exprimées sous forme de **délais** compris entre la survenance des événements d'une classe E1 et celle des événements d'une autre classe E2 successeurs des premiers dans le schéma spécifié (cf. I.5.) ;
2. une **estimation quantitative** du comportement soit disponible. Cette estimation se fait sur base de **mesures de fonctionnement,** réel ou simulé, du comportement du système d'information spécifié. Ces **mesures de fonctionnement** doivent permettre non seulement d'évaluer les performances du système en fonctionnement mais aussi de **localiser et d'expliquer** les problèmes, c'est-à-dire les divergences entre les performances attendues et obtenues.

Dans le contexte de l'environnement logiciel adopté, ces mesures de fonctionnement correspondent aux **résultats statistiques** obtenus à l'aide de l'**outil d'évaluation par simulation** présenté au chapitre précédent (cf. chapitre II.).

La démarche proposée pour évaluer la faisabilité des spécifications de comportement est la suivante :

1. **Spécification** à l'aide du langage proposé, de la solution retenue - comme **hypothèse** de travail. Cette spécification comprend les trois aspects suivants :
   - comportement fonctionnel ;
   - prise en compte des moyens requis ;
   - et expression de la charge estimée ;
2. **Contrôle de la cohérence** des spécifications telle que développée dans la section précédente ;
3. Spécification des **performances attendues** de la solution retenue et spécifiée ;
4. Génération automatique et exécution d'une **simulation** ;

5. **Exploitation des résultats statistiques** produits par la simulation. Elle devrait permettre la détection des anomalies dans le fonctionnement du système spécifié qui rendent la solution retenue difficilement acceptable ou même réalisable par l'organisation.

Au terme de cette évaluation, l'hypothèse peut s'avérer :
- acceptable et rendre compte d'une solution faisable ;
- inacceptable parce que traduisant un fonctionnement pathologique dont les anomalies invalident plus ou moins sérieusement l'hypothèse retenue. Dans ce cas, il convient d'ajuster les spécifications et de réitérer la procédure développée ci-dessus.

En fonction de l'exploitation des résultats statistiques qui aurait mis en évidence un comportement pathologique ou simplement parce qu'on souhaiterait tester une autre hypothèse alternative, on pourrait en effet être amené à ou vouloir modifier :

1. la **charge estimée** en adaptant les échéanciers d'événements initiaux en conséquence ;
2. les **moyens requis** ou ressources à mettre en oeuvre :
   - soit en ajustant leur calendrier de disponibilité
   - soit en introduisant d'autres ressources et en ajustant les réquisitions par les processus ;
3. le **comportement fonctionnel** proprement dit en restructurant les traitements (processus) et donc leurs enchaînements (événement, survenance et déclenchement).

L'examen des résultats statistiques peut enfin entraîner une remise en cause des performances attendues initialement spécifiées et un ajustement de ces dernières en conséquence.

L'outil d'évaluation par simulation favorise donc une **approche expérimentale** par "essais-erreurs". Il permet en effet d'envisager différentes hypothèses alternatives et de construire progressivement des spécifications garantissant l'efficience du système d'information conçu et à réaliser.

### III.4.2. Contrôle de la faisabilité par interprétation des résultats statistiques de simulation

La démarche suivante peut servir de canevas général pour l'**interprétation des résultats statistiques** d'une simulation en vue de **localiser et d'expliquer** les éventuelles anomalies de fonctionnement dans le

comportement d'un système d'information. Cette démarche comprend 4 étapes dont les objectifs respectifs sont de détecter :

1. les divergences entre performances attendues et obtenues ;
2. les retards anormaux dans le déclenchement de processus ;
3. les attentes anormales de processus déclenchés ;
4. les ressources critiques.

La détection de ces quatre types de problèmes se fait par l'interprétation des résultats statistiques (cf. II.4.) qui concernent respectivement :

1. la survenance des événements,
2. le déclenchement des processus,
3. l'évolution des processus et
4. le comportement des ressources.

En fonction de la précision recherchée dans la localisation et l'explication des comportements critiques ou pathologiques, cette interprétation peut se faire globalement, périodiquement et/ou chronologiquement par un ajustement approprié des paramètres des outils logiciels d'exploitation statistique (cf. II.4.3.).

### III.4.2.1. **Détection des divergences entre performances attendues et obtenues**

Cette étape a pour objectif d'identifier et de **localiser les divergences** entre les **performances attendues** telles qu'elles ont été spécifiées (cf. I.5.) et les **performances estimées** par simulation.
La détection de ces divergences est effectuée par l'interprétation des **résultats statistiques concernant la survenance des événements** (cf. II.4.2.1.).

En effet, la spécification d'une performance attendue établit précisément les valeurs limites de la durée comprise entre les dates de survenances de deux événements successeurs dans un schéma. Il suffit dès lors de **comparer,** pour les deux classes dont les événements se succèdent et sur lesquelles porte la spécification de performance, les **délais spécifiés et ceux estimés** par simulation. Ce délai estimé correspond aux résultats statistiques concernant la **durée inter-événements.** Dans la mesure où une durée inter-événements entre un événement E1 et un événement E2 qui lui succède ne peut évidemment être dégagée qu'à la survenance d'un événement E2, il convient donc également d'examiner le **nombre de survenances** établi pour les deux classes d'événements considérées.

Exemple III.10.

Pour la spécification d'une performance attendue établie de la façon suivante :

DEF EVENEMENT commande-préparée
SUIT commande-du-jour DE moins-de-24H

on dégagerait les résultats statistiques suivants concernant la durée inter-événements pour les classes d'événements "commande du jour" et "commande préparée" :

Résultats statistiques concernant la survenance des événements "commande préparée"
sur la période du 5ème au 20ème jour de la simulation

Nombre de survenances : 375

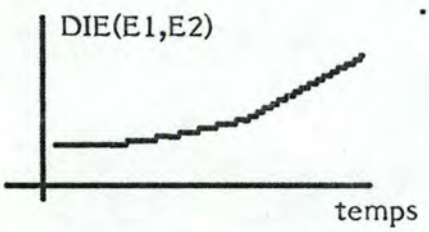
| Durée inter-événements | minimum | moyenne | maximum |
|---|---|---|---|
| "commande du jour" | 4:30:17 | 21:43:05 | 48:16:39 |

Nombre de survenances de "commande du jour" : 400

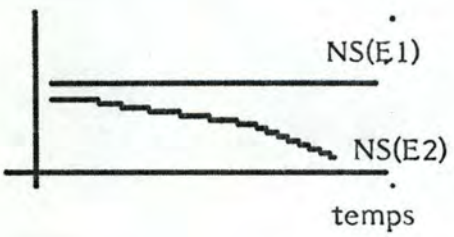
La comparaison entre la performance attendue et les statistiques concernant la durée inter-événements correspondante permettrait de constater qu'en moyenne les résultats obtenus par simulation (21:43:05) sont conformes aux spécifications ("moins de 24H).

Toutefois, le fait que le nombre de survenances des "commandes préparées" (375) soit inférieur à celui des "commandes du jour" (400) et que la valeur maximale de la durée inter-événements (48:16:39) soit assez élevée devrait attirer l'attention et justifier une analyse complémentaire des résultats statistiques concernés.

Cette analyse plus détaillée consisterait notamment à examiner l'évolution dans le temps de la durée inter-événements et des nombres de survenances. On pourrait, par exemple, par une analyse plus chronologique d'un phénomène, détecter une croissance anormale de la durée inter-événements au fur et à mesure de la progression dans le temps, comme schématiquement illustré à la figure suivante :




De la même façon, on pourrait détecter une divergence de plus en plus accrue entre les nombres de survenances d'événements E1 et d'événements E2 successeurs :



### III.4.2.2. Détection des retards anormaux dans le déclenchement de processus

Cette étape a pour objectif d'**identifier,** pour la partie du schéma où l'on a détecté une divergence entre performances attendues et obtenues, les causes d'éventuels **retards dans le déclenchement** de certains processus. L'identification de ces causes de retard correspond en fait à la détection des classes d'événements - dits contraignants - qui surviennent tardivement et retardent le déclenchement des processus.

La détection de ces survenances tardives, qui traduisent généralement une filière critique, est effectuée par l'interprétation des **résultats statistiques concernant le déclenchement des processus** (cf. II.4.2.2).

En effet, l'examen des statistiques concernant la durée de déclenchement permet d'identifier les classes de processus pour lesquelles les délais de déclenchement semblent anormalement longs. Pour ces classes de processus au déclenchement jugé trop tardif, l'analyse peut se poursuivre par un examen des statistiques qui concernent la **durée de contribution** des événements contribuants sachant qu'un **événement** est d'autant **plus contraignant** qu'il contribue tardivement au déclenchement, c'est-à-dire que **sa durée de contribution est courte.**

Si cette interprétation a permis de détecter une classe d'événements contraignants qui en plus est une classe d'événements internes (non initiaux), il s'agira de poursuivre l'analyse des résultats en la centrant principalement sur la filière critique identifiée, c'est-à-dire sur la partie du schéma précédant - en amont de - cette classe d'événements.

La détection évoquée ci-dessus n'a bien sûr de sens qu'en cas de synchronisation, c'est-à-dire quand la survenance d'au moins deux événements est nécessaire pour déclencher un processus. Dans le cas contraire, lorsque la précondition d'un processus n'exprime pas un phénomène de synchronisation, la durée de déclenchement des processus est toujours nulle et aucun retard ne peut fatalement être détecté, ni même exister.

Exemple III.11.

La spécification d'un comportement, dont la simulation a révélé qu'il ne respectait pas les performances attendues, établit notamment qu'un processus "enregistrement livraison" est déclenché en cas de survenance d'un événement "contrôle-qualité-terminé" et d'un autre "contrôle-quantité-terminé". Dans ce cas, on produirait les résultats statistiques suivants concernant le déclenchement des processus "enregistrement livraison" :

Résultats statistiques concernant le déclenchement des processus "enregistrement livraison"
pour la période du 5ème au 20ème jour de la simulation

Nombre de déclenchements : 345

| Durée de déclenchement | minimum | moyenne | maximum |
|---|---|---|---|
| | 00:12:34 | 17:12:00 | 34:52:69 |
| Durée de contribution pour | | | |
| (345) contrôle-qualité-terminé | - | 1:17:24 | 00:42:00 |
| (345) contrôle-quantité-terminé | - | 12:02:13 | 34:52:69 |

La simple lecture de ce rapport devrait mettre en évidence que la classe d'événements "contrôle-qualité-terminé" est plus contraignante que celle des "contrôle-quantité-terminé" puisqu'en moyenne un événement "contrôle-quantité-terminé" doit attendre 18H qu'un événement "contrôle-qualité-terminé" survienne pour déclencher un processus "enregistrement livraison". Cette constatation devrait justifier une analyse complémentaire des résultats centrée sur l'examen du comportement en amont de la classe d'événements "contrôle-qualité-terminé".

### III.4.2.3. Détection des attentes anormales de processus - déclenchés

Cette étape a pour objectif de localiser, spécialement dans une partie critique du schéma détectée lors des étapes précédentes, les classes de processus dont les **attentes** sont **anormalement longues** et/ou les interruptions trop fréquentes ou trop longues.

La détection de ces classes de processus critiques est effectuée par l'interprétation des **résultats statistiques concernant l'évolution des processus** (cf. II.4.2.3.).

Dans un premier temps, le seul examen des nombres caractérisant l'évolution des processus d'une même classe permet déjà de s'assurer que tous les processus déclenchés ont pu démarrer, redémarrer s'ils ont été interrompus et enfin qu'ils se sont terminés.

Dans un second temps, l'examen des statistiques portant sur les durées caractéristiques de l'évolution des processus peut mettre en évidence des attentes ou interruptions anormalement longues dont il conviendra d'expliquer les causes par une analyse des ressources requises telle que développée à l'étape suivante.

Enfin, l'analyse des statistiques associées aux volumes de processus dans un certain état caractéristique permet de détecter des volumes anormaux de processus en attente ou interrompus à un moment donné dont il s'agira également d'identifier les causes.

Exemple III.12.

Supposons qu'au terme d'une simulation l'évolution des processus "enregistrement commande" soit caractérisée par les résultats statistiques suivants :

Résultats statistiques concernant l'évolution des processus "enregistrement commande"
sur la période du 5ème au 20ème jour de la simulation

| | |
|---|---|
| Nombre de déclenchements | 9472 |
| démarrages | 8148 |
| interruptions | 192 |
| redémarrages | 174 |
| terminaisons | 7891 |

| | minimum | moyenne | maximum |
|---|---|---|---|
| Durée d'attente | 0:00:00 | 1:54:47 | 4:59:29 |
| interruption | 0:00:00 | 0:52:03 | 2:00:00 |
| activité | 0:03:00 | 0:08:00 | 0:13:00 |
| totale d'exécution | | 2:54:50 | |
| Volume en "attente" | 0 | 290.3 | 621 |
| "en interruption" | 0 | 10 | 58 |
| "en activité" | 0 | 10.2 | 15 |
| "en cours d'exécution" | | 304.5 | |

L'examen de ces résultats devrait attirer l'attention et vraisemblablement justifier une analyse complémentaire pour découvrir la cause des durées d'attente et - dans une moindre mesure - d'interruption assez importantes (près de 2H) comparées à la durée d'activité des processus (8 min. en moyenne).

Cette analyse complémentaire devrait permettre non seulement de découvrir les ressources critiques dont la - trop - forte sollicitation explique les attentes anormales mais aussi de suivre l'évolution des résultats sur des périodes plus fines pour éventuellement détecter des périodes plus critiques que d'autres.

### III.4.2.4. **Détection des ressources critiques**

Cette étape a pour objectif principal d'identifier les **ressources critiques** dont la **trop forte utilisation** ou la **trop faible capacité disponible** justifie les attentes et interruptions intempestives ou anormalement longues détectées à l'étape précédente.

Toutefois sachant que les performances attendues d'un système peuvent être obtenues au prix d'une très - trop - forte utilisation des ressources ou au contraire moyennant une sous-utilisation - anti-économique - des ressources, il peut s'avérer utile de passer en revue toutes les ressources et d'en examiner le comportement même en l'absence de problèmes particuliers détectés par ailleurs.

La détection de ces ressources critiques est effectuée par l'interprétation des **résultats statistiques concernant le comportement des ressources** (cf. II.4.2.4.).

En particulier, l'examen des volumes de requêtes dans un état donné et du taux d'utilisation des ressources requises, par exemple, par les processus d'une classe jugée critique, doit normalement permettre la détection assez aisée des ressources critiques ; une ressource étant jugée critique si sa trop forte utilisation et/ou capacité maximale disponible l'empêchent de faire face à la demande des processus et entraînent des files en attente importantes de requêtes.

Exemple III.13.

Sachant que chaque processus "enregistrement commande" requiert un poste de travail du "service réception" et l'attention du "chef de service", les résultats statistiques suivant pourraient être analysés pour détecter la ressource critique :

Résultats statistiques (moyens) concernant le comportement des ressources requises par les processus "enregistrement commande"
sur la période du 5ème au 20ème jour de la simulation

| | service réception | chef de service |
|---|---|---|
| volumes des requêtes | | |
| en attente | 700.1 | 700.1 |
| en cours d'utilisation | 28.3 | 56.4 |
| en cours de séjour | 728.4 | 756.5 |
| taux d'utilisation | 91 % | 47 % |

L'examen de ces résultats très synthétiques mettrait assez clairement en évidence que - globalement - la forte utilisation (91 %) de la ressource "service réception" semble plus expliquer les retards des processus analysés que la ressource "chef de service" dont l'utilisation semble plus normale.

Toutefois, comme pour les étapes précédentes, l'examen de ces résultats globaux - valeur moyenne pour toute la période de référence considérée - devrait être complété par une analyse plus fine visant à appréhender l'évolution des phénomènes sur des périodes plus fines comprises dans la période de référence. Sur base de résultats statistiques produits par période ou chronologiquement, il serait alors possible de détecter les périodes pendant lesquelles la ressource semble plus critique qu'à d'autres moments (en fin ou début de semaine, par exemple).

*
* *

# BIBLIOGRAPHIE

ALFORD M.W. ; A Requirements Engineering Methodology for Real - Time Processing Requirements ; IEEE Transactions on Software Engineering, Vol. SE-3, n° 1, January 1981.

ATZENI P., BATINI C., DE ANTONELLIS V., LENZERINI M., VILLANELLIS F. and ZONTA B. ; A Computer-Aided Tool for Conceptual Data Base Design ; in H.S. Schneider and A.I. Wasserman (eds.), Automated Tools for Information Systems Design, IFIP, North-Holland, 1982.

BARROW J. ; Dialog and Process Design for Interactive Information Systems using TAXIS ; Technical Report CSRG-128, Computer Systems Research Group, University of Toronto, April 1981.

BAUDOIN M., BRIAND H., WALLE J.M. ; Méthode de détermination des traitements ; Séminaire Inforsid II, Aix-en-Provence, 15-16 janvier 1980.

BELL T.E., BIXLER D.C. and DYER M.E. ; An Extendable Approach to Computer - Aided Software Requirements Engineering ; IEEE Transactions on Software Engineering, Vol. SE-3, n° 1, January 1977.

BENCI E., BODART F., BOGAERT H. and CABANES A. ; Concepts for the Design of a Conceptual Schema ; in C.M. Nijssen (ed.), Modelling in DBMS, IFIP, North-Holland, 1976.

BEYERLE W. ; A Simulation Generator for Easier Evaluation of Computer Systems ; SIMULATION'77, June 1977.

BLOOM T. ; Synchronization Mechanisms for Modular Programming Languages ; MIT/LCS/TR-211, Massachusetts Institute of Technology, 1979.

BODART F. ; Logical Specification for System Dynamics - An Extension to PSL ; ISDOS Working Paper n° 183, University of Michigan, April 1977.

BODART F. and PIGNEUR Y. ; A Model and Language for Fonctional Specification and Evaluation of System Dynamics ; Proceedings of the IFIP - TC8 WC on Formal Methods and Practical Tools for Information System Design, Oxford, North-Holland, 1977.

BODART F. and PIGNEUR Y. ; Logical Specification for System Dynamics : an Extension to PSL ; European ISDOS Meeting, Namur, June 1978.

BODART F. and PIGNEUR Y. ; Problem Dynamics Specification and Information System Simulation ; US ISDOS Project Review Meeting, Ann Arbor, September 1978.

BODART F. et PIGNEUR Y. ; Spécifications dynamiques des systèmes d'information ; Séminaire Inforsid II, Cergy-Pontoise, octobre 1978.

BODART F. et PIGNEUR Y. ; Modèle de simulation d'un système d'information ; Séminaire Inforsid II, Aix-en-Provence, janvier 1980.

BODART F., HENNEBERT A.M., LEHEUREUX J.M. and PIGNEUR Y. ; Dynamics Specification Language and a Simulation Model Analyzer for Information System ; US ISDOS Project Review Meeting, Ann Arbor, August 1980.

BODART F. ; Eléments de conception et d'analyse des systèmes d'information ; Ecole d'été - AFCET, Thies (Sénégal), juillet 1981.

BODART F. et LESUISSE R. ; Une méthode d'analyse opérationnelle en informatique d'organisation ; FNDP, Research Paper 2/82, 1982.

BODART F., HENNEBERT A.M., LEHEUREUX J.M., MASSON O. and PIGNEUR Y. ; DSL-DSA : A System for Requirements Specification, Prototyping and Simulation ; Proceedings of the IFIP-TC2 WC on System Description Methodologics, Kecskement, Hungary, North-Holland, 1983.

BODART F. et PIGNEUR Y. ; Conception assistée des applications informatiques : Etude d'opportunité et analyse conceptuelle ; Masson, 1983.

BOYDSTON L.E., TEICHROEW D., SPEWAK S., YAMAMOTO Y. and STARNER G. ; Computer-Aided Modelling of Information Systems ; COMPSAC Proceedings of IEEE Fourth International Conference on Computer Software and Applications, October 1980.

BOYDSTON L.E., TEICHROEW D., GAUTHIER S.G. and ALLEN S.S. ; Dynamics Analysis of Information Systems - Requirements using SDL/SDA ; ISDOS MO533-O, University of Michigan, 1983.

BRACCHI G., FURTADO A. and PELAGATTI G. ; Constraint Specification in Evolutionary Data Base Design ; in H.S. Schneider (ed.), Formal Models and Practical Tools for Information System Design, Proceedings of IFIP TC-8 WC, Oxford, North-Holland, 1979.

BRAMS G.W. ; Réseaux de Petri : Théorie et Pratique ; Masson, 1983.

BREUTMANN B., FALKENBERG E. and MAUER R. ; CSL : A Language for Defining Conceptual Schemas ; in IFIP WC on Data Base Architecture, North-Holland, 1979.

BRODIE M.L. and SILVA E. ; Active and Passive Component Modelling : ACM/PCM ; in T.W. Olle, H.G. Sol, A.A. Verrijn-Stuart (eds.), Information Systems Design Methodologies, IFIP, North-Holland, 1982.

BUKENKO J. ; The Temporal Dimension in Information Modelling ; RC 6187 # 26479, IBM Research Center, YorkTown, New York, 1979.

CAMPBELL R.H. and HABERMANN N. ; The Specification of Process Synchronization by Path Expressions ; Lecture Notes in Computer Science, Vol. 16, Springer-Verlag, 1974.

CHEN P.P. ; The Entity-Relationship Model. Toward an Unified view of Data ; ACM Transactions on Data Base Systems, Vol. 1, n° 1, 1976.

COMPUTER (IEEE) ; Special Issue on Programming Environments ; Vol. 14, n° 4, April 1981.

DAHLE O.J., MYHRHAUG B., NYGAARD K. ; SIMULA67 Common Base Language ; Pub. S-22, Norwegian Computer Center, 1970.

DAHLE O.J., PIENE J. ; Evaluation of Computer Systems through Simulation ; Management Informatics, Vol. 2, n° 6, 1973.

DAVIS A.M. and RATAJ W.J. ; Requirements Language Processing for the Effective Testing of Real-Time Systems ; Proceedings of Software Quality and Assurance Workshop, SIGSOFT, Vol. 3, n° 5, 1978.

DAVIS C.G. and VICK C.R. ; The Software Development System ; IEEE Transactions on Software Engineering, Vol. SE-3, n° 1, January 1977.

DE ANTONELLIS V., ZONTA B. ; Modelling Events in Data Base Application Design ; in Proceedings of International Conference on very large Data Bases, Cannes, 1981.

de DROUAS E. et NERSON J.M. ; Les ateliers logiciels intégrés : développements français actuels ; TSI, Vol. 1, n° 3, mai-juin 1982.

DE MARCO ; Structured Analysis and System Specification ; a Yourdon Book, Prentice-Hall, 1979.

DUFOURD J.F. ; Maquettes pour évaluer les systèmes d'information des organisations ; Thèse d'Etat, Université de Nancy, 1980.

ELLIS C.A. ; Information Control Nets : A Mathematical Model of Office Information Flow ; Proceedings of ACM/SIGSIM Conference on Simulation, Measurement and Modelling of Computer Systems, Boulder, SIGSIM Vol. 11, n° 1, Fall, 1979.

ELLIS C. and NUTT G. ; Office Information Systems and Computer Science ; ACM Computing Survey, Vol. 12, n° 1, March 1980.

FICHEFET J. ; Introduction à la simulation ; Ecole d'été d'Informatique de l'AFCET, Lannion, 1976.

FISHMAN G. ; Concepts and Methods in Discrete Event Digital Simulation ; Wiley, 1973.

FLORY A. et KOULOUMDJIAN J. ; La représentation des règles d'évolution d'un système d'information ; Colloque sur les Bases de Données-Modèles, Mise en oeuvre, Evaluation, Paris, 1979.

FLORY A. and KOULOUMDJIAN J. ; A Model for the Description of the Information System Dynamics ; Séminaire Inforsid II, Cergy-Pontoise, 2-3 octobre 1978.

FOUCAUT O. and ROLLAND C. ; Concepts for Design of an Information System Conceptual Schema and its Utilization in the REMORA Project ; Proceedings of Fourth Conference on very large Data Bases, Berlin, 1978.

FOUCAUT O. ; Modèle et Outil pour la conception des systèmes d'information dans les organisations ; Thèse de doctorat, Université de Nancy, 1982.

FORRESTER J.W. ; Industrial Dynamics ; Mit Press, 1961.

FRANKOWSKI E. and FRANTA W.R. ; A Process Oriented Simulation Model Specification and Documentation Language ; Software-Practice and Experience, Vol. 10, 1980.

FRANTA W.R. ; The Process view of Simulation ; The Computer Science Library, North-Holland, 1977.

FRANTA W.R. and MALY K. ; An Efficient Data Structure for the Simulation Event Set ; Communications of the ACM, Vol. 20, n° 8, August 1977.

GOLDMAN N.M., WILE D.S. ; A Data Base Foundation for Processes Specifications ; Report ISI/RR-80-84, USC-Information Sciences Institute, 1980.

GUSTAFSON M.R., KARLSON T. and BUKENKO J.A. ; A Declarative Approach to Conceptual Information Modelling ; in T.W. Olle, H.G. Sol, A.A. Verrijn-Stuart (eds.), Information Systems Design Methodologies, IFIP, North-Holland, 1982.

HAINAUT J.L. ; Conception de bases de données - Notes de Cours ; FNDP Institut d'Informatique, 1984.

HAMMER M., HOWE W.G., HRUSHAL V.J. and WLADAWSKY I. ; A very High Level Programming Language for Data Processing Applications ; Communications of the ACM, Vol. 20, n° 11, November 1977.

HARRISON P.G. ; The Finite State Machine as a Software Engineering Tool ; IBM Research Center, RC 8861 (# 38784), San Jose, May 1981.

HECKENROTH H., TARDIEU H. et ESPINASSE B. ; Modèles et outils pour la conception de la cinématique d'un système d'information ; Rapport CETE, Aix-en-Provence, 1980.

HOARE C.A.R. ; Communicating Sequential Processes ; Communications of the ACM, Vol. 21, n° 8, August 1978.

HOLBACK-HANSEN, HANDLYKKEN and NYGAARD ; Systems Description and the DELTA language ; DELTA Projekt report n° 4, Norwegian Computing Center, Oslo, February 1977.

HSU J. and ROUSSOPOULOS N. ; Data Base Conceptual Modelling, in P. Chen (ed.), Proceedings of International Conference on the E-R Approach to Systems Analysis and Design, North-Holland, 1979.

HÜNKE H. (ed.) ; Software Engineering Environments ; North-Holland, 1980.

HUTCHINSON G.K. ; Introduction to the Use of Activity Cycles as a Basis for System's Decomposition and Simulation ; SIMULETTER, Vol. 11, n° 1, October 1975.

KOBAYASHI H. ; Modelling and Analysis : an Introduction to System Performance Evaluation Methodology ; Addison-Wesley, 1978.

IIVARI J. ; Contributions to the Theoritical Foundations of Systemeering Research and the Pioco Model ; Acta Universitatis Ouluensis, A150, Cybern. 1, Ovlu, Finland, 1983.

KAMPEN G.R. ; SWIFT : A Requirements Specification System for Software ; in Y. Ohno (ed.), International Symposium on current Issue of Requirements Engineering Environments, OHMSHA LTD, 1982.

KONSYNSKI B.R. ; A Model of Computer-Aided Definition and Analysis of Information System Requirements ; Ph.D dissertation, Purdue University, December 1975.

KONSYNSKI B.R. ; Macro Simulation in Design of Information System ; Proceedings Ninth Hawaii International Conference on Systems Sciences, Western Periodicals, January 1976.

KRAKOWIAK S. ; Systèmes intégrés de production de logiciels : concepts et réalisations ; TSI, Vol.1, n° 3, mai-juin 1982.

KREUTZER W. ; Evaluation of Computer System Simulation Tools ; Computer Performance Evaluation, ONLINE Conference Limited, England, 1976.

LANDWEHR C.E. ; An Abstract Type for Statistics Collection in SIMULA67 ; ACM Transactions on Programming Languages and Systems, Vol. 2, n° 4, October 1980.

LATTEUX M. ; Synchronisation de Processus ; RAIRO-Informatique, Vol. 14, n° 2, 1980.

LEONARD M. et LUONG B.T. ; Approche de la conception des systèmes d'information intégrant les données et les traitements ; Séminaire Inforsid II, Cergy-Pontoise, 2-3 octobre 1978.

LEVESON N.G. and WASSERMAN A.I. ; BASIS : A Behavioral Approach to the Specification of Information Systems ; Inform. Systems, Vol. 8, n° 1, 1983.

LEVESQUE H. and MYLOPOULOS J. ; A Procedural Semantics for Semantic Networks ; in N.V. Findler (ed.), Associative Networks, Academic Press, 1979.

LINDGREEN P. ; Event Directed Program Execution from Declarative Specifications ; Proceedings of the IFIP-TC8 WC on Formal Methods and Practical Tools for Information System Design, Oxford, North-Holland, 1977.

LUCAS H.C., LAND F.F., LINCOLN T.J. and SUPPER K. (eds.) ; The Information Systems Environment ; Proceedings of the IFIP - TC8.2 WC on The Information Systems Environment, Bonn, North-Holland, 1979.

LUDEWIG J. ; A Process Control Specification Language ; ISDOS Working Paper 283, University of Michigan, September 1979.

MAC DOUGALL M.H. ; Computer System Simulation : An Introduction ; ACM Computing Survey, Vol. 2, n° 3, September 1970.

MADDISON R.N. ; Information System Methodologies ; British Computer Society Monographs in Informatics, Wiley Heyden, 1983.

MAGER P.S. ; A Possible Approach for Specifying Performance Characteristics in PSL ; 2nd East Coast PSL/PSA User's conference, New Port, November 1982.

Mc LEOD D. ; A Semantic Data Base Model and its Associated Structured User Interface ; MIT-LCS-TR-214, Massachusetts Institute of Technology, 1978.

MEYER B. and DEMUYNCK M. ; Specification Languages : A Critical Survey and Proposal ; IEEE/ACM Conference on Specification of Reliable Software, Cambridge (US), 1979.

MILLER E. ; Tutorial on Automated Tools for Software Engineering ; COMPSAC 79, IEEE Computer Society Press, 1979.

MITTERMEIR R.T., HSIA P. and YEH R.T. ; Alternatives to Overcome the Communication Problem of Formal Requirements Analysis ; in Y. OHNO (ed.) Proceedings of the International Symposium on Current Issues of Requirements Engineering Environments, Kyoto, Japan, September 1982.

NAUMANN J.D. and JENKINS M.A. ; Prototyping : The New Paradigm for Systems Development ; MIS quarterly, Vol. 6, n° 3, September 1982.

NOE J.D. and NUTT G.J. ; Macro E-Nets for Representation of Parallel Systems ; IEEE Transactions on Computers, Vol. C-22, n° 8, August 1973.

NUNAMAKER V.F. and KONSYNSKI B.R. ; Computer-Aided Analysis and Design of Information Systems ; Communications of the ACM, Vol. 19, n° 12, December 1976.

NUTT G.J. ; Evaluation Net Simulation System-Reference Manual ; University of Colorado, Boulder, TR # CU-CS-042-74, April 1974.

OHNO Y. (ed.) ; Requirements Engineering Environments ; Proceedings of the International Symposium on Current Issues of Requirements Engineering Environments, Kyoto, North-Holland, 1982.

OLLE T.W., SOL H.G. and TULLY C.J. (eds.) ; Information Systems Design Methodologies : A Feature Analysis ; Proceedings of the IFIP WG8.1 WC on Feature Analysis of Information Systems Design Methodologies, York (U.K.), North-Holland, 1981.

OLLE T.W., SOL H.G. and VERRIJN-STUART A.A. (eds.) ; Information Systems Design Methodologies : A Comparative Review ; Proceedings of the IFIP WG8.1 WC on Comparative Review of Information Systems Design Methodologies, Noordwijkerhout, North-Holland, 1982.

OSTERWEIL L. ; Software Environment Research : Directions for the Next Five Years ; IEEE Computer, Vol. 14, n° 4, April 1981.

PALME J. ; Structured Simulation Programming ; Pub. C10082-M3 (E5), Norwegian Computer Center, March 1978.

PEAUCELLE J.L. ; La durée de l'événement ; Séminaire Inforsid II, Cergy-Pontoise, 2-3 octobre 1978.

PETERSON J.L. ; Petri Nets ; ACM Computing Survey, Vol. 9, n° 3, Septembre 1977.

PRITSKER A.A. ; Modelling and Analysis using Q-GERT Networks ; J. Wiley, New York, 1977.

PRITSKER A.A. ; The GASP IV Simulation Language ; Wiley, 1974.

RAMAMRITHAM K. and KELLER R.M. ; Specification of Synchronizing Processes ; IEEE Transactions on Software Engineering, Vol. SE-9, n° 6, November 1983.

RAZOUK R., VERNON M. and ESTRIN G. ; Evaluation Methods in SARA - The Graph Model Simulator ; Proceedings of ACM/SIGSIM Conference on Simulation Measurement and Modelling of Computer Systems, Boulder, SIGSIM Vol. 11, n° 1, Fall 1979.

REED D.P. and KANODIA R.K. ; Synchronization with Event Counts and Sequencers ; Communications of the ACM, Vol. 22, n° 2, February 1979.

REUVENI A. ; The Event Based Language and its Multiple Processor Implementations ; Ph. D dissertation, MIT-LCS-TR229, Massachusetts Institute of Technology, 1980.

RIDDLE W.E., WILEDEN J.C., SAYLER J.H., SEGAL A.R. and STAVENY A.M. ; Behavior Modelling During Sotware Design ; IEEE Transactions on Software Engineering, Vol SE-4, n° 4, July 1978.

RIDDLE W.E. and SAYLER J.H. ; Modelling and Simulation in the Design of Complex Software Systems ; in B.P. Zeigler and others (eds.), Methodology in Systems Modelling for Simulation, North-Holland, 1979.

RIDDLE W.E. and FAIRLEY R.E. ; Software Development Tools ; Springer-Verlag, New York, 1980.

ROLLAND C., LEIFFERT S. et RICHARD C. ; Le pilotage de l'évolution des systèmes d'information ; Séminaire Inforsid II, Aix-en-Provence, 15-16 janvier 1980.

ROLLAND C. and RICHARD C. ; The Remora Methodology for Information Systems Design and Management ;
in OLLE T.W., SOL H.G. and VERRIJN-STUART A.A. (eds.), Information Systems Design Methodologies : A Comparative Review, North-Holland, 1982.

ROSS D.T. ; Structured Analysis (SA) : A Language for Communicating Ideas ; IEEE Transactions on Software Engineering, Vol. SE-3, n° 1, January 1977.

ROUCAIROL G.P. ; Mots de synchronisation ; RAIRO-Informatique, Vol. 12, n° 4, 1978.

SANGUINETTI J. ; A Technique for Integrating Simulation and System Design ; Conference on simulation Measurement and Modelling of Computer System, Boulder, ACM/SIGSIM, Vol. 11, n° 1, Fall, 1979.

SCHNEIDER H.J. and WASSERMAN A.I. (eds.) ; Automated Tools for Information Systems Design ; North-Holland, New York, 1982.

SCHRIBER T.J. ; Simulation using GPSS ; J. Wiley, New York, 1974.

SCHUMAKER F. ; The S-Net System user's Manual ; Kernforschungszentrune Karlsruhe Gmbtt, Report n° GFK-IDT/ 263-02, November 1976.

SERNADAS A. ; Temporal Aspects of Logical Procedure Definition ; in Schneider (ed.), Information Systems, Vol. 5, 1980.

SOL H.G. ; Simulation in Information Systems Development (Ph. D) ; Rijksuniversiteit te Groningen, 1982.

SPEWAK S.H. ; Analysis of Dynamics of the Logical Design of Information Systems ; Ph. D dissertation, University of Michigan, 1980.

SPRIET J.A. and VANSTEENKISTE G.C. ; Computer-Aided Modelling and Simulation ; Academic Press, 1982.

TARDIEU H., ROCHFELD A. et COLLETTI R. ; La méthode MERISE ; les éditions d'organisation, 1983.

TEICHROEW D. ; Use of PSL/PSA in the Development and Maintenance of Real-Time Systems ; ISDOS Technical Memorandum 71, December 1977.

TEICHROEW D. and HERSHEY III E.A. ; PSL/PSA : A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems ; IEEE Transactions on Software Engineering, Vol. SE-3, n° 1, January 1977.

TEICHROEW D. and SPEWAK S. ; Research in Simulation of Information Systems - Guidelines for Research by the ISDOS Project ; ISDOS Technical Memorandum 73, University of Michigan, February 1978.

TEICHROEW D., MACASOVIC P., HERSHEY III E. and YAMAMOTO Y. ; Application of the Entity - Relationship Approach to Information Processing Systems Modelling ; in P.P. Chen (ed.), Entity-Relationship Approach to System Analysis and Design, North-Holland, 1980.

TSICHRITZIS D. ; Omega Alpha ; Technical Report CSRG-127, University of Toronto, March 1981.

van LAMSWEERDE A. ; Les outils d'aide au développement de logiciels : un aperçu des tendances actuelles ; XVèmes Journées Internationales de l'Informatique et de l'Automatisme, Paris, 1982.

van LAMSWEERDE A. ; Automatisation de la production de logiciels ; TSI, Vol. 1, n° 6-8, 1982.

VAUCHER J. ; Programmation de simulation ; Université de Montréal, Montréal, 1971.

VAUCHER J. and DUVAL P. ; A Comparison of Simulation Event List Algorithms ; Communications of the ACM, Vol. 18, n° 4, April 1975.

WASSERMAN A.I. ; The Use Software Engineering Methodology : An Overview ; Proceedings of the IFIP TC-8 Working Conference on Comparative Review of Information Systems Design Methodologies, Noordwijkerhout, The Netherlands, North-Holland, May 1982.

WASSERMAN A.I. ; Tutorial : Software Development Environments ; COMPSAC 81, IEEE Computer Society Press, 1981.

WILLIS R.R. ; A Man/Machine Wokload Model ; ACM Simuletter, Vol. 8, n° 1, October 1976.

WILLIS R.R. ; DAS : An Automated System to Support Design Analysis ; Proceedings Third International Conference on Software Engineering, Atlanta, May 1978.

WINCHESTER J. ; Design Analysis System ; ISDOS Project Review Meeting, Ann Arbor, September 1977.

WINCHESTER J. ; Requirements Definition and its Interface to the SARA Design Methodology for Computer-Based Systems ; Ph. D dissertation, University of California, 1980.

WINCHESTER D.L., DESHLER M.D. and WINCHESTER J.W. ; Integrated Software Development Tools and Techniques for Real-Time Systems ; ISDOS Project Review Meeting, University of Michigan, 1983.

WYMAN F.P. ; Improved Event-Scanning Mechanisms for Discrete Event Simulation ; Communications of the ACM, Vol. 18, n° 6, June 1975.

YAMAMOTO Y. ; An Approach to the Generation of Software Life Cycle Support Systems ; Ph. D dissertation, University of Michigan, Ann Arbor, 1982.

ZAVE P. ; An Operational Approach to Requirements Specification for Embedded Systems ; IEEE Transactions on Software Engineering, Vol. SE-8, n° 3, May 1982.

ZEIGLER B.P. ; Theory of Modelling and Simulation ; Wiley, 1976.

ZEIGLER B.P., ELZAS M.S., KLIR G.J. and OREN J.I. (eds.) ; Methodology in Systems Modelling and Simulation ; North-Holland, 1979.

ZISMAN M.D. ; Use of Production Systems for Modelling Asynchronous Concurrent Processes ; in Pattern Directed Inference Systems, Academic Press, 1978.

## ANNEXE : EXEMPLE D'UNE GESTION - SIMPLIFIEE - DE PRETS

Un organisme financier souhaite mettre en place un système d'information, partiellement automatisé, de gestion des prêts qu'il octroie à ses clients.

Cet exemple illustre une démarche possible pour spécifier fonctionnellement le comportement - ou dynamique - de ce système d'information à l'aide du modèle et des outils présentés dans ce travail.

L'élaboration progressive de ces spécifications se fera selon la démarche suivante :

1. Spécification générale du comportement ;
2. Spécification - plus fine - du comportement fonctionnel proprement dit ;
3. Vérification de la complétude ;
4. Vérification de la cohérence interne ;
5. Vérification de la cohérence externe ;
6. Spécification d'une hypothèse de moyens ;
7. Vérification de la faisabilité ;
8. Version DSL des spécifications et des rapports statistiques.

### 1. Spécification générale du comportement

Lorsqu'une demande de prêt est introduite, elle est traitée et donne lieu à une acceptation (cf. I.2.) :

```
DEF PROCESSUS        Gestion-de-prêt
   DECLENCHE PAR     Demande-de-prêt
   CAUSE   1 Prêt-accepté SI Critères-satisfaits,
           1 Prêt-refusé  SI Critères-non-satisfaits
```

Les responsables de la gestion des prêts - les futurs utilisateurs du système - souhaitent que le système conçu fournisse des performances (cf. I.5.), telles qu'une demande de prêt soit traitée et donc que la réponse soit expédiée au client moins de 24 heures après l'introduction de la demande :

```
DEF EVENEMENTS       Prêt-accepté, Prêt-refusé
   SUCCEDE A         Demande-de-prêt APRES Environ-24-heures
```

Enfin, l'examen de la situation actuelle a permis d'estimer la charge (cf. I.4.) du futur système qui devrait pouvoir absorber environ 40 demandes de prêts par jour :

```
DEF EVENEMENT      Demande-de-prêt
   SURVIENT        1 Fois TOUTES LES Environ-12-min PENDANT
                   Heures-d'ouverture-clientèle

   DEF PARAMETRE   Environ-12-min
      DISTRIBUTION NEGEXP AVEC PARAMETRE 12 min

   DEF PERIODE     Heures-d'ouverture-clientèle
      DE 8H 30 min  A 11H 30 min,
         13H 30 min  A 16H 30 min DANS Semaine-de-travail

      DEF PERIODE Semaine-de-travail
         DE 0 A 5 jours DANS SEMAINE
```

Cette première spécification - fort globale - peut être affinée car le traitement d'une demande de prêt supppose l'exécution d'un certain nombre d'activités dont le comportement fonctionnel sera précisé au point suivant de cette annexe. La spécification suivante ne reprend que le nom de ces différentes activités (cf. I.6.) :

```
DEF PROCESSUS      Gestion-de-prêt
   SOUS-PROCESSUS SONT Enregistrement-demande-de-prêt,
                       Préparation-dossier,
                       Examen-crédit, Décision-crédit,
                       Consolidation-dossier,
                       Acceptation-du-prêt, Refus-du-prêt
```

**2. Spécification plus détaillée du comportement fonctionnel proprement dit (cf. I.2.)**

Une demande de prêt, émanant d'un client qui se présente au service des prêts, fait l'objet d'un enregistrement qui a pour objectif d'ouvrir un dossier et d'y incorporer les renseignements indispensables au traitement ultérieur de la demande :

```
DEF PROCESSUS      Enregistrement-demande-de-prêt
   SYNONYME        Ouverture-dossier
   DECLENCHE PAR   Demande-de-prêt
   CAUSE 1 Demande-de-prêt-enregistrée
```

Lorsque la demande de prêt est enregistrée, deux traitements peuvent être exécutés en parallèle. D'une part, le dossier peut être complété et préparé

en vue d'une éventuelle acceptation du prêt (calcul des mensualités,...) :

```
DEF PROCESSUS Préparation-dossier
   DECLENCHE PAR Demande-enregistrée
   CAUSE 1 Dossier-préparé
```

D'autre part, il s'agit d'examiner le crédit du client pour s'assurer qu'il satisfait aux conditions d'obtention du prêt :

```
DEF PROCESSUS      Examen-crédit
   DECLENCHE PAR Demande-enregistrée
   CAUSE 1 Crédit-examiné
```

Suite à cet examen du crédit, qui correspond à la préparation de la décision, la prise de décision proprement dite peut avoir lieu :

```
DEF PROCESSUS      Décision-crédit
   DECLENCHE PAR Crédit-examiné
   CAUSE 1 Crédit-décidé
```

Lorsque le dossier a été préparé et la décision de crédit prise, le dossier est consolidé et provisoirement clôturé. Au terme de ce traitement, la demande de prêt est soit acceptée, soit refusée par le service des prêts :

```
DEF PROCESSUS      Consolidation-dossier
   DECLENCHE PAR Dossier-préparé, Crédit-décidé
                 QUAND Dossier-à-consolider
   CAUSE 1 Demande-acceptée SI Critères-satisfaits
         1 Demande-refusée  SI Critères-non-satisfaits

   DEF CONDITION Dossier-à-consolider
     PREDICAT :
                 N°-dossier DE Dossier-préparé =
                 N°-dossier DE Crédit-décidé

   DEF CONDITION Critères-satisfaits
     PREDICAT :
                 VRAIE-DANS (80 %)

   DEF CONDITION Critères-non-satisfaits
     PREDICAT :
                 Critères-satisfaits EST FAUX
```

Il reste alors à informer le client de la décision prise et de lui signaler, par courrier, si le prêt lui est ou non accordé :

DEF PROCESSUS Acceptation-du-prêt
DECLENCHE PAR Demande-acceptée
CAUSE 1 Prêt-accepté

DEF PROCESSUS Refus-du-prêt
DECLENCHE PAR Demande-refusée
CAUSE 1 Prêt-refusé

**3. Vérification de la complétude (cf. III.2.)**

Une fois les spécifications ci-dessus enregistrées dans la base de données de spécification, tous les contrôles de complétude prévus au chapitre 3 (cf. III.2.2.) peuvent être automatiquement effectués à l'aide du langage d'interrogation.

Les spécifications respectent la plupart de ces règles de complétude. Toutefois, certains critères ont quand même entraîné la sélection et l'extraction d'informations de la base de données révélant parfois des spécifications incomplètes. Seuls les critères non satisfaits sont repris ci-dessous :

a. PROCESSUS AND NOT (HAS PROCEDURE) :

Acceptation-du-prêt
Consolidation-dossier
Décision-crédit
Enregistrement-demande-de-prêt (SYNONYME ouverture-dossier)
Examen-crédit
Gestion-de-prêt
Préparation-dossier
Refus-du-prêt

Toutes les classes de processus ont été sélectionnées puisque leur procédure n'a pas - encore - été spécifiée. Les spécifications sont donc incomplètes. Toutefois, le non-respect de cette règle n'empêche pas le contrôle de la cohérence et de la faisabilité des spécifications.

b. EVENEMENT AND NOT (? DECLENCHE !) :

Prêt-accepté
Prêt-refusé

Il s'agit en fait des deux classes d'événements terminaux. Il est donc normal qu'elles aient été sélectionnées.

## 4. Vérification de la cohérence interne (cf. III.3.2.)

Une approche par visualisation a été retenue pour vérifier l'absence d'incohérence dans les spécifications du comportement fonctionnel décrites au point 2 de cette annexe. La démarche exposée au chapitre 3 (cf. III.3.2.1.) a donc été appliquée de la façon suivante :

a. Sélection des classes d'événements terminaux à l'aide du langage d'interrogation :

```
EVENEMENT AND ? CAUSE PAR ! AND.NOT (? DECLENCHE !) :
        Prêt-accepté
        Prêt-refusé
```

b. Production automatique du rapport de structuration restituant les classes d'événements et de processus précédant les événements des deux classes sélectionnées ci-dessus :

```
1   1 Prêt-accepté                              EVT
2    2 Acceptation-du-prêt                      PRC
3     3 Demande-acceptée                        EVT  (Critères-satisfaits)
4      4 Consolidation-du-dossier               PRC  (Dossier-à-consolider)
5       5 Dossier-préparé                       EVT
6        6 Préparation-du-dossier               PRC
7         7 Demande-enregistrée                 EVT
8          8 Enregistrement-demande-de-prêt     PRC
9           9 Demande-de-prêt                   EVT
10      5 Crédit-décidé                         EVT
11       6 Décision-crédit                      PRC
12        7 Crédit-examiné                      EVT
13         8 Examen-crédit                      PRC
14          9 Demande-enregistrée               EVT  (*)

    CONDITIONS :
        Critères-satisfaits : VRAIE-DANS (80 %)
        Dossier-à-consolider : N°-dossier DE Dossier-préparé =
                               N°-dossier DE Crédit-décidé

15  1 Prêt-refusé                               EVT
16   2 Refus-du-prêt                            PRC
17    3 Demande-refusée                         EVT  (Critères-non-satifaits)
18     4 Consolidation-du-dossier               PRC  (*)

    CONDITIONS :
        Critères-non-satisfaits : Critères-satisfaits FAUX
```

c. L'analyse - la simple lecture - du rapport met en évidence que, dans les deux cas, la règle d'accessibilité est vérifiée puisque
   - la dernière classe sélectionnée "Demande-de-prêt" (ligne 9) est bien une classe d'événements initiaux ;
   - aucune contradiction ne peut être détectée dans les conditions de déclenchement et de survenance reprises dans le rapport.

**5. Vérification de la cohérence externe (cf. III.3.3.)**

Il s'agit de vérifier que les spécifications du comportement fonctionnel reprises au point 2 de cette annexe respectent celles données au point 1 et donc
- qu'à **un** événement "Demande-de-prêt" (Précondition de "Gestion-de-prêt")
- correspond ou succède **un** événement "Prêt-accepté" **ou un** événement "Prêt-refusé" (Postcondition de "Gestion-de-prêt").

La correspondance au niveau des noms de classes peut facilement - car automatiquement - être vérifiée par la procédure d'interrogation détaillée au chapitre 3 (cf. III.3.3.1.) :

```
evt-décl-GP = EVENEMENT AND ? DECLENCHE Gestion-de-prêt ;
evt-causé-par-GP = EVENEMENT AND ? CAUSE PAR Gestion-de-prêt ;
ss-proc-GP = PROCESSUS AND ? SOUS-PROCESSUS DE Gestion-de-prêt ;
evt-init = EVENEMENT AND ? DECLENCHE ss-proc-GP
                    AND NOT (? CAUSE PAR ss-proc-GP) ;
evt-term = EVENEMENT AND ? CAUSE PAR ss-proc-GP
                    AND NOT (? DECLENCHE ss-proc-GP) ;
evt-décl-GP AND NOT evt-init :        NONE
evt-init    AND NOT evt-décl-GP :     NONE
evt-causé-par-GP AND NOT evt-term :   NONE
evt-term AND NOT evt-causé-par-GP :   NONE
```

Toutefois, il reste encore à vérifier l'absence de contradiction dans les modalités de déclenchement et de survenance et donc le respect des pré- et post-conditions des processus "Gestion-de-prêt" par les spécifications détaillées du comportement fonctionnel. Une approche par exécution a été retenue et la procédure de contrôle correspondante (cf. III.3.3.2.) appliquée de la façon suivante :

a. La précondition des processus "Gestion-de-prêt" (1 Demande-de-prêt) a été traduite par la spécification d'un échéancier ne prévoyant la survenance que d'un seul événement :

```
DEF EVENEMENT  Demande-de-prêt
   SURVIENT 1 FOIS
```

b. Une simulation a été générée automatiquement au départ de "Demande-de-prêt" et exécutée ;

c. La "trace" suivante a été produite en cours de simulation :

```
0:00:00 survenance de          Demande-de-prêt      [INITIAL]
0:00:00 déclenchement de       Ouverture-dossier
0:00:00 survenance de          Demande-enregistrée
0:00:00 déclenchement de       Préparation-dossier
0:00:00 déclenchement de       Examen-crédit
0:00:00 survenance de          Dossier-préparé
0:00:00 *** condition Dossier-à-consolider non vérifiée
0:00:00 survenance de          Crédit-examiné
0:00:00 déclenchement de       Décision-crédit
0:00:00 survenance de          Crédit-décidé
0:00:00 déclenchement de       Consolidation-dossier (Dossier-à-consolider VRAIE)
0:00:00 survenance de          Demande-acceptée (Critères-satisfaits VRAIE)
0:00:00 déclenchement de       Acceptation-du-prêt
0:00:00 survenance de          Prêt-accepté         [TERMINAL]
0:00:00 FIN DE LA SIMULATION
```

L'analyse de cette "trace" permet de vérifier que les spécifications ne contredisent pas la postcondition des processus "Gestion-de-prêt" puisqu'un événement terminal "Prêt-accepté" succède bien à un événement initial "Demande-de-prêt" (= Précondition) sans qu'un événement terminal "Prêt-refusé" ne lui succède.

Une seconde simulation, pour laquelle on forcerait la condition "Critères-satisfaits" à FAUX, pourrait également se justifier pour s'assurer que la survenance d'un événement "Prêt-refusé" exclut celle d'un événement "Prêt-accepté".

**6. Spécification d'une hypothèse de moyens (cf. I.3.)**

Le service des prêts est constitué de 10 stations ou postes de travail, qui correspondent à 10 employés disposant chacun d'un terminal. L'horaire de travail des employés de ce service correspond aux heures d'ouverture des bureaux :

```
DEF RESSOURCE     Station-service-des-prêts
   CAPACITE 10 DISPONIBLE PENDANT Heures-d'ouverture

   DEF PERIODE    Heures-d'ouverture
      DE 8H 30min    A  12H 30min,
         13H 30 min  A  17H 30min  DANS Semaine-de-travail
```

Ces stations ou postes de travail peuvent indifféremment assurer l'ouverture, la préparation et la consolidation d'un dossier. L'ouverture et la consolidation d'un dossier demandent entre 20 et 40 minutes à une quelconque de ces stations :

```
DEF PROCESSUS Ouverture-dossier, Consolidation-dossier
   REQUIERT 1 Station-service-des-prêts
   ACTIF PENDANT Entre-20-et-40 min

   DEF DOMAINE-DE-VALEURS Entre-20-et-40 min
     VALEURS ENTRE 20 min ET 40 min
```

La préparation d'un dossier, par contre, ne dure qu'entre 10 et 20 minutes :

```
DEF PROCESSUS  Préparation-dossier
   REQUIERT 1 Station-service-des-prêts
   ACTIF PENDANT Entre-10-et-20 min

   DEF DOMAINE-DE-VALEURS Entre-10-et-20 min
     VALEURS ENTRE 10 min ET 20 min
```

L'examen et la décision de crédit sont effectués par les employés d'un autre service. Le service du crédit comprend 7 employés qui ont les mêmes heures de travail que le service des prêts :

```
DEF RESSOURCE Employé-service-du-crédit
   CAPACITE 7 DISPONIBLE PENDANT Heures-d'ouverture
```

L'examen du crédit pour un dossier donné est une activité qui occupe un employé pendant une durée très variable selon la complexité du dossier :

```
DEF PROCESSUS  Examen-du-crédit
   REQUIERT 1 Employé-service-du-crédit
   ACTIF PENDANT Entre-10-et-200 min

   DEF DOMAINE-DE-VALEURS Entre-10-et-200 min
     VALEURS ENTRE 10min ET 20min AVEC PROBABILITE DE 20 %,
             ENTRE 21min ET 60min AVEC PROBABILITE DE 50 %,
             ENTRE 61min ET 90min AVEC PROBABILITE DE 20 %,
             ENTRE 91min ET 200min AVEC PROBABILITE DE 10 %
```

La décision d'accorder ou non le prêt est prise par un des deux directeurs du service de crédit en présence d'un des employés qui lui soumet le dossier. Cette activité de décision leur prend environ 5 minutes :

```
DEF PROCESSUS  Décision-crédit
   ACTIF PENDANT Entre-2-et-8 min
   REQUIERT 1 Directeur-service-du-crédit
            1 Employé-service-du-crédit

   DEF DOMAINE-DE-VALEURS Entre-2-et-8 min
     VALEURS ENTRE 2 min ET 8 min
```

Les deux directeurs ne consacrent cependant qu'une partie de leur temps aux demandes de prêts et ne sont disponibles qu'une heure le matin et l'après-midi :

```
DEF RESSOURCE  Directeur-service-du-crédit
   CAPACITE 2 DISPONIBLE PENDANT Horaire-direction

   DEF PERIODE Horaire-direction
     DE 11H A 12H,
        15H A 16H DANS Semaine-de-travail
```

De plus, dans la mesure où le temps que les directeurs accordent à l'étude des dossiers est limité, il est acquis qu'un examen de crédit est moins prioritaire qu'une décision de crédit et même qu'un employé qui effectue un examen peut s'interrompre pour assister à une décision de crédit :

```
DEF PROCESSUS  Examen-crédit
   MOINS PRIORITAIRE QUE Décision-Crédit
   INTERRUPTION : AUTORISEE
```

Enfin, l'expédition d'une décision, positive ou négative, au client prend 2 minutes sur l'imprimante du service expédition qui est disponible 2 heures par jour :

```
DEF PROCESSUS  Acceptation-du-prêt, Refus-du-prêt
   ACTIF PENDANT 2 min
   REQUIERT 1 Imprimante-service-expédition

   DEF RESSOURCE  Imprimante-service-expédition
     CAPACITE 1 DISPONIBLE PENDANT Heures-d'expédition

     DEF PERIODE  Heures-d'expédition
       DE 12H A 13H,
          17H A 18H DANS Semaine-de-travail
```

## 7. Vérification de la faisabilité (cf. III.4.)

La vérification de la faisabilité de la solution spécifiée aux points 1 et 2 de cette annexe en fonction de l'hypothèse de moyens retenue et décrite au point 6 suppose que la complétude et la cohérence des spécifications, incluant celles du point 6 (les ressources), aient été préalablement vérifiées. Cette vérification est semblable à celle effectuée aux points 3, 4 et 5 de cette annexe et utilise les mêmes procédures (cf. III.2. et III.3.).

L'analyse de la faisabilité se fait sur base de mesures de fonctionnement qui peuvent être obtenues par simulation. L'outil logiciel d'évaluation a donc été utilisé et une simulation, correspondant aux spécifications précédentes, automatiquement générée et exécutée (sur une période d'une semaine).

Une exploitation des résultats statistiques, ayant pour objectif de détecter les éventuelles anomalies dans le comportement du futur système, a dès lors pu être menée selon la démarche - en 4 temps - proposée et détaillée au chapitre 3 (cf. III.4.2.) :

a. Détection des divergences entre performances attendues et obtenues (cf. III.4.2.1.)

D'une part, les spécifications de performances prévoient (point 1 de cette annexe) qu'une demande de prêt soit examinée et suivie d'une acceptation ou d'un refus du prêt dans les 24 heures.

D'autre part, on dispose des résultats statistiques suivants :

Résultats statistiques concernant la survenance des événements "Demande-de-prêt"
pour la période du 1er au 5ème jour de la simulation

Nombre de survenances (résultats établis sur 194 observations)

| | |
|---|---|
| Demande-de-prêt | 204 |
| Prêt-accepté | 160 |
| Prêt-refusé | 34 |

| Durée inter-événements | minimum | moyenne | maximum |
|---|---|---|---|
| Prêt-accepté | 54m | 10H44m | 29H22m |
| Prêt-refusé | 54m | 10H49m | 29H14m |

L'étude de ce rapport permet de constater qu'en moyenne l'exigence de performance (< 24 heures) est satisfaite (10H 44-49min) pour 194 (160 + 34) demandes de prêts sur les 204 qui ont été introduites. Toutefois, la valeur maximum du délai (29H 14-22min) doit attirer l'attention et justifie l'analyse complémentaire effectuée ci-dessous pour tâcher de localiser et expliquer ce retard.

b. Détection des retards anormaux dans le déclenchement de processus (cf. III.4.2.2.)

Seul le déclenchement des processus "Consolidation-dossier" fait l'objet d'une condition de synchronisation et nécessite donc une analyse des résultats statistiques correspondants :

Résultats statistiques concernant le déclenchement des processus "Consolidation-dossier"
pour la période du 1er au 5ème jour de la simulation
Nombre de déclenchements : 194

| | minimum | moyenne | maximum |
|---|---|---|---|
| Durée de déclenchement : | 2m | 7H59m | 20H48m |
| Durée de contribution pour | | | |
| (194) Dossier-préparé | 0 | 7H59m | 20H48m |
| (194) Crédit-décidé | 0 | 10s | 15m |

L'étude de ce rapport montre qu'en moyenne il faut presque 8 heures avant qu'un processus "Consolidation-dossier" ne soit déclenché, c'est-à-dire que deux événements (un dossier-préparé et un crédit-décidé) soient survenus pour contribuer à son déclenchement. Mais cette étude met également en évidence que les événements contraignants sont les événements "Crédit-décidé" puisqu'en moyenne ils surviennent avec 8 heures de retard par rapport aux événements "Dossier-préparé".

Ce retard dans le déclenchement couvrant pratiquement 75 % du temps total de traitement d'une demande de prêt, il convient dès lors d'examiner plus particulièrement, dans la suite de l'exploitation des résultats, la filière "Examen-" et "Décision-crédit" pour tâcher d'expliquer ce retard systématique.

c. Détection des attentes anormales de processus (cf. III.4.2.3.)

Pour tâcher de localiser et d'expliquer le retard de la filière "Examen-" et "Décision-crédit", les deux rapports suivants ont été demandés et produits :

Résultats statistiques concernant l'évolution des processus "Examen-crédit"
pour la période du 1er au 5ème jour de la simulation

Nombre de déclenchements : 204
démarrages : 204
interruptions : 48
redémarrages : 45
terminaisons : 201

| | minimum | moyenne | maximum |
|---|---|---|---|
| Durée d'attente | 0 | 4m | 43m |
| Durée d'interruption | 0 | 2m | 17m |
| Temps mort (cf. II.4.2.3.) | 0 | 1H10m | 15H |
| Durée d'activité | 10m | 47m | 3H12m |
| Durée totale d'exécution | 10m | 2H3m | 19H27m |
| Volume en attente | 0 | .38 | 5 |
| interruption | 0 | .02 | 7 |
| activité | 0 | 4.02 | 7 |
| en cours d'exécution | | 4.42 | |

Résultats statistiques concernant l'évolution des processus "Décision-crédit"
pour la période du 1er au 5ème jour de la simulation

Nombre de déclenchements : 201
démarrages : 194
terminaisons : 194

| | minimum | moyenne | maximum |
|---|---|---|---|
| Durée d'attente | 0 | 24m | 1H22m |
| Temps mort | 0 | 5H27m | 19H12m |
| Durée d'activité | 2m | 5m | 8m |
| Durée totale d'éxécution | 2m | 5H56m | 20H42m |
| Volume en attente | 0 | 7.88 | 31 |
| activité | 0 | 1.54 | 2 |
| en cours d'exécution | | 9.42 | |

L'étude comparative de ces deux rapports permet de constater que la durée totale d'exécution des processus "Décision-crédit" (5H56m) explique pour 75 % le retard dans le déclenchement des processus "Consolidation-dossier" (7H59m) alors que leur durée d'activité (5m) est cependant plus courte que celle des processus "Examen-crédit" (47m).

Cette constatation que les processus "Décision-crédit" semblent davantage critiques que les processus "Examen-crédit" est également confirmée par le volume des processus "Décision-crédit" en attente (7.88) plus élevé que celui des processus "Examen-crédit" (.38).

Il convient donc d'examiner plus particulièrement les ressources requises par les processus "Décision-crédit" dans la localisation, effectuée à l'étape suivante de la démarche, des ressources critiques du système d'information spécifié.

Avant d'entamer l'étape suivante et afin de mieux localiser les phénomènes d'attente constatés au niveau des processus "Décision-Crédit", les résultats statistiques ont également été demandés **périodiquement** heure par heure d'un journée moyenne de simulation :

Résultats statistiques **périodiques** concernant l'évolution des processus "Décision-crédit"

Heure par heure d'une journée moyenne de la simulation :

| | | décl. | Nombres de démar. | Nombres de term. | Durée Attente | Temps mort |
|---|---|---|---|---|---|---|
| 8H | 9H | 12 | | | | |
| 9H | 10H | 17 | | | | |
| 10H | 11H | 25 | | | | |
| 11H | 12H | 26 | 109 | 107 | 25m | 8H48m |
| 12H | 13H | 9 | | | | |
| 13H | 14H | 5 | | | | |
| 14H | 15H | 27 | | | | |
| 15H | 16H | 37 | 85 | 87 | 23m | 1H16m |
| 16H | 17H | 30 | | | | |
| 17H | 18H | 13 | | | | |

L'examen de ce rapport montre clairement que les démarrages n'ont lieu qu'entre 11 heures et midi et entre 15 heures et 16 heures alors que les déclenchements sont répartis dans la journée. Seule l'indisponibilité d'une ou plusieurs ressources peut expliquer ce phénomène. L'étape suivante de la démarche proposée aura donc pour but de détecter la ou les ressources critiques dont l'indisponibilité temporaire explique le comportement de ces processus.

d. Détection des ressources critiques (cf. III.4.2.4.)

Sachant que chaque processus "Décision-crédit" requiert un employé et un directeur du service de crédit, le rapport statistique suivant à été

demandé :

Résultats statistiques (moyens) concernant le comportement des ressources requises par les processus "Décision-crédit"
pour la période du 1er au 5ème jour de la simulation

| | Employé-service-du-crédit | Directeur-service-du-crédit |
|---|---|---|
| Nombre de réquisitions | 453 | 201 |
| acquisitions | 443 | 194 |
| libérations | 443 | 194 |
| Volume des requêtes | | |
| en attente | 10.41 (2.53) | 7.88 |
| en cours d'utilisation | 4.38 | 1.54 |
| en cours de séjour | 14.79 | 9.42 |
| Taux d'utilisation | 62 % | 78 % |
| Calendrier de disponibilité (SPECIFICATIONS) | | |
| Capacité | 7 | 2 |
| Période | 8:30-12:30<br>13:30-17:30 | 11:00-12:00<br>15:00-16:00 |

L'étude de ce dernier rapport met en évidence que la faible disponibilité des 2 directeurs (2 heures par jour) explique principalement les attentes anormales des requêtes de processus "Décision-crédit". En effet, d'une part, ces deux directeurs ont un taux d'utilisation assez élevé (78 %) pendant leurs 2 heures de prestation. D'autre part, le nombre de requêtes en attente (donc de processus "Décision-crédit" bloqués) reste également élevé (7.88) sur toute la période simulée.

Par contre, les employés du service ont non seulement un taux d'utilisation plus raisonnable (62 %) mais aussi un volume réel de requêtes en attente plus faible (2.53). Ce dernier chiffre (2.53) correspond en fait à la différence du volume total de requêtes en attente d'un employé (10.41) et du volume de requêtes en attente d'un directeur (7.88) car à chaque réquisition d'un directeur correspond également celle d'un employé.

L'exploitation des résultats statistiques faite ci-dessus pourrait bien sûr être prolongée par des investigations complémentaires pour préciser les constatations élaborées. De la même façon, sur base des remarques effectuées

et des réactions qu'elles pourraient susciter de la part des utilisateurs, on pourrait être amené à établir et évaluer une nouvelle hypothèse de moyens (réexaminer, par exemple, les calendriers de disponibilité). Toutefois, la spécification et la vérification de cette nouvelle solution se feraient selon la même démarche que celle abondamment illustrée dans cet exemple.

## 8. Version DSL des spécifications et des rapports statistiques

La dernière partie de cette annexe donne l'expression DSL (1) des spécifications précédentes ainsi que certains exemples de rapports statistiques produits par DSL-SIM (2).

---

(1) DSL est le langage de spécification, en version anglaise, actuellement disponible dans l'environnement logiciel d'IDA (cf. préambule).

(2) DSL-SIM est la version actuellement fournie par IDA de l'outil intégré d'évaluation par simulation présenté au chapitre II.

Input Processor Source Listing

Parameters: LANGUAGE=dsl DB=PRET.DBF INPUT=PRET.DSL SOURCE-LISTING
CROSS-REFERENCE UPDATE DATA-BASE-REFERENCE

```
 1 * exemple de GESTION DE PRETS
 2 * (version DSL)
 3
 4 * Comportement fonctionnel (proprement dit)
 5 *
 6 DEF PROCESS Enreg-demande-pret;
 7    SYNONYMS Ouverture-dossier;
 8    TRIGGERED ON GENERATION OF Demande-de-pret;
 9
10 DEF PROCESS Preparation-dossier;
11    TRIGGERED ON TERMINATION OF Ouverture-dossier;
12
13 DEF PROCESS Examen-credit;
14    TRIGGERED ON TERMINATION OF Ouverture-dossier;
15
16 DEF PROCESS Decision-credit;
17    TRIGGERED ON TERMINATION OF Examen-credit;
18
19 DEF PROCESS Consolidation-dossier;
20    TRIGGERED BY Dossier-a-consolider;
21
22 *  modalites de declenchement
23
24    DEF SYNCHRONIZATION-POINT Dossier-a-consolider;
25       CONTRIBUTED BY TERMINATION OF Decision-credit, Preparation-dossier;
26       REALIZED-WHEN;
27          AND
28             TERM-OF(Decision-credit)
29             TERM-OF(Preparation-dossier) ;
30
31 DEF PROCESS Acceptation-du-pret;
32    TRIGGERED ON TERMINATION OF Consolidation-dossier IF Pret-accepte;
33
34 DEF PROCESS Refus-du-pret;
35    TRIGGERED ON TERMINATION OF Consolidation-dossier IF NOT Pret-accepte;
36
37 *  modalites de survenance
38
39    DEF CONDITION Pret-accepte;
40       PROBABILITY TRUE IS 0.80;
41
42 * Charge estimee
43 *
44 DEF MESSAGE Demande-de-pret;
45    HAPPENS EVERY Environ-12-minutes FROM Clients DURING Horaire-clients;
45
47    DEF SYSTEM-PARAMETER Environ-12-minutes;
48       DISTRIBUTION IS NEGEXP WITH PARAMETER '12m';
49
50 * Moyens requis
```

Input Processor Source Listing

```
  51 *
  52 DEF RESOURCE Station-service-des-prets;
  53     CAPACITY IS 10;
  54     AVAILABLE DURING Horaire-de-travail;
  55
  56     DEF PROCESS Ouverture-dossier;
  57         PERFORMED DURING Entre-20-et-40-minutes;
  58         REQUIRES 1 Station-service-des-prets;
  59
  60         DEF SYSTEM-PARAMETER Entre-20-et-40-minutes;
  61             RANGE IS '20m' THRU '40m';
  62
  63     DEF PROCESS Preparation-dossier;
  64         PERFORMED DURING Entre-10-et-20-minutes;
  65         REQUIRES 1 Station-service-des-prets;
  66
  67         DEF SYSTEM-PARAMETER Entre-10-et-20-minutes;
  68             RANGE IS '10m' THRU '20m';
  69
  70     DEF PROCESS Consolidation-dossier;
  71         PERFORMED DURING Entre-20-et-40-minutes;
  72         REQUIRES 1 Station-service-des-prets;
  73
  74         DEF SYSTEM-PARAMETER Entre-20-et-40-minutes;
  75             RANGE IS '20m' THRU '40m';
  76
  77 DEF RESOURCE Employe-service-credit;
  78     CAPACITY IS 7;
  79     AVAILABLE DURING Horaire-de-travail;
  80
  81 DEF RESOURCE Directeur-service-credit;
  82     CAPACITY IS 2;
  83     AVAILABLE DURING Horaire-direction;
  84
  85     DEF PROCESS Examen-credit;
  86         PERFORMED DURING Entre-10-et-200-minutes;
  87         REQUIRES 1 Employe-service-credit;
  88
  89         DEF SYSTEM-PARAMETER Entre-10-et-200-minutes;
  90             RANGE IS '10m' THRU '20m'  WITH PROBABILITY 0.20;
  91             RANGE IS '21m' THRU '60m'  WITH PROBABILITY 0.50;
  92             RANGE IS '61m' THRU '90m'  WITH PROBABILITY 0.20;
  93             RANGE IS '91m' THRU '200m' WITH PROBABILITY 0.10;
  94
  95     DEF PROCESS Decision-credit;
  96         PREEMPTIVE PRIORITY 1;
  97         PERFORMED DURING Entre-2-et-8-minutes;
  98         REQUIRES 1 Directeur-service-credit;
  99         REQUIRES 1 Employe-service-credit;
 100
 101         DEF SYSTEM-PARAMETER Entre-2-et-8-minutes;
 102             RANGE IS '2m' THRU '8m';
 103
```

Input Processor Source Listing

```
104 DEF RESOURCE Imprimante-service-expedition;
105     CAPACITY IS 1;
106     AVAILABLE DURING Horaire-expedition;
107
108     DEF PROCESS Acceptation-du-pret;
109         REQUIRES 1 Imprimante-service-expedition;
110         PERFORMED DURING '2m';
111
112     DEF PROCESS Refus-du-pret;
113         REQUIRES 1 Imprimante-service-expedition;
114         PERFORMED DURING '2m';
115
116 # Periodes (calendriers)
117 #
118 DEF CALENDAR Horaire-clients;
119     ACTIVE Mat-cli, Apr-mid-cli ON Sem-trav;
120
121     DEF HRS-INTERVAL Mat-cli;
122         SPANS FROM '8h30m' TO '11h30m';
123
124     DEF HRS-INTERVAL Apr-mid-cli;
125         SPANS FROM '13h30m' TO '16h30m';
126
127 DEF CALENDAR Horaire-de-travail;
128     ACTIVE Mat-trav, Apr-mid-trav ON Sem-trav;
129
130     DEF HRS-INTERVAL Mat-trav;
131         SPANS FROM '8h30m' TO '12h30m';
132
133     DEF HRS-INTERVAL Apr-mid-trav;
134         SPANS FROM '13h30m' TO '17h30m';
135
136 DEF CALENDAR Horaire-direction;
137     ACTIVE Mat-direct, Apr-mid-direct ON Sem-trav;
138
139     DEF HRS-INTERVAL Mat-direct;
140         SPANS FROM '11h' TO '12h';
141
142     DEF HRS-INTERVAL Apr-mid-direct;
143         SPANS FROM '15h' TO '16h';
144
145 DEF CALENDAR Horaire-expedition;
146     ACTIVE Mat-exped, Apr-mid-exped ON Sem-trav;
147
148     DEF HRS-INTERVAL Mat-exped;
149         SPANS FROM '12h' TO '13h';
150
151     DEF HRS-INTERVAL Apr-mid-exped;
152         SPANS FROM '17h' TO '18h';
153
154 DEF DAY-INTERVAL Sem-trav;
155     SPANS FROM 0 TO 5 DAY PER WEEK;
```

Structured Formated Statement Report

Parameters: LANGUAGE=dsl DB=PRET.DBF NAME=Demande-de-pret NOPUNCH
OBJECTS=ALL RELATIONS=DWN-DYNAMICS PRINT INDENT=2 LEVELS=50
TYPES-MARGIN=36 RELATIONS-MARGIN=48 LINE-NUMBERS LEVEL-NUMBERS
STATISTICS NONEW-PAGE OBJECT-TYPES ROW-ORDER=STANDARD
COLUMN-ORDER=STANDARD NOLOWEST-LEVEL-ONLY NORELATION-MATRIX
NOCOMPRESS-RELATION-MATRIX

```
1   1 Demande-de-pret              <MESSAGE>
2     2  Enreg-demande-pret        <PROCESS>
                                       TRIGGERED BY GENERATION OF
                                       Demande-de-pret
3       3   Preparation-dossier   <PROCESS>
                                       TRIGGERED BY TERMINATION OF
                                       Enreg-demande-pret
4        4    Dossier-a-consolider<SYNCH-POINT>
                                       CONTRIBUTED BY TERMINATION OF
                                       Preparation-dossier
5         5     Consolidation-dossier
                                 <PROCESS>
                                       TRIGGERED BY REALIZATION OF
                                       Dossier-a-consolider
6          6      Acceptation-du-pret
                                 <PROCESS>
                                       TRIGGERED BY TERMINATION OF
                                       Consolidation-dossier IF
                                       Pret-accepte
7          6      Refus-du-pret  <PROCESS>
                                       TRIGGERED BY TERMINATION OF
                                       Consolidation-dossier IF NOT
                                       Pret-accepte
8       3   Examen-credit         <PROCESS>
                                       TRIGGERED BY TERMINATION OF
                                       Enreg-demande-pret
9        4    Decision-credit     <PROCESS>
                                       TRIGGERED BY TERMINATION OF
                                       Examen-credit
10         5     Dossier-a-consolider
                                 <SYNCH-POINT> *
                                       CONTRIBUTED BY TERMINATION OF
                                       Decision-credit
```

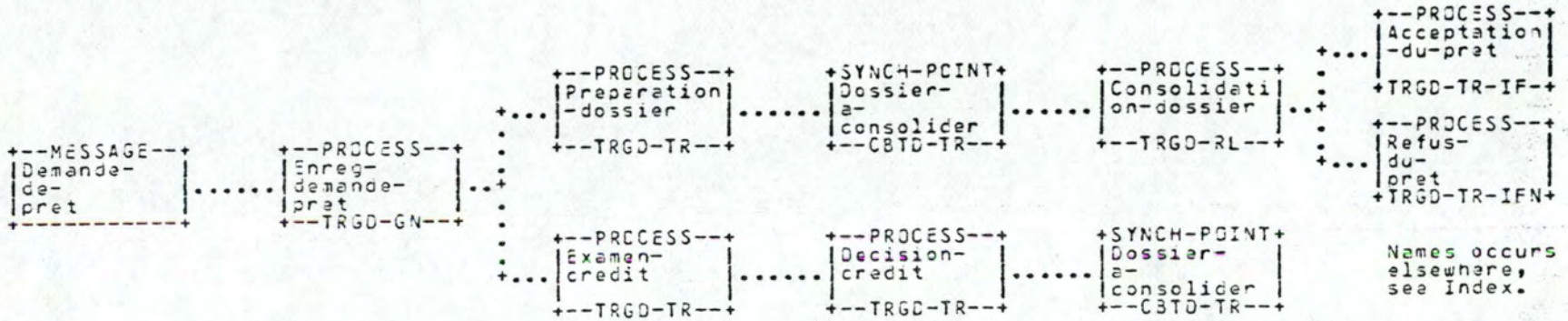
| LEVEL | COUNT | LEVEL | COUNT | LEVEL | COUNT | LEVEL | COUNT | LEVEL | COUNT |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 3 | 2 | 4 | 2 | 5 | 2 |
| 0 | 2 | | | | | | | | |

FNDP - Namur VAX/VMS

Extended Picture

NAME=Demande-de-pret






Names occurs elsewhere, see Index.

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Acceptation-du-pret

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Acceptation-du-pret
SPANS FROM 12h TO 13h, FROM 17h TO 18h ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 160 |
| Inception Number | 160 |
| Termination Number | 160 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 2m | 2m | 2m | 0s |
| Waiting Time | 0s | 16m51s | 53m60s | 13m60s |
| Idle Time by Incp/Rsmp | 0s | 50m9s | 3h29m41s | 42m39s |
| Idle Time by Process | | 50m9s | | |
| Elapse Time | | 1h8m59s | | |
| Numb of Proc in Wt State | 0.00 | 4.57 | 27.00 | 5.69 |
| Numb of Proc in Actv State | 0.00 | 0.54 | 1.00 | 0.50 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Imprimante-service-expedition | 1.00 | 0.66 | | 0.66 | 5.58 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 11h | 12h | 39 | 0 | 0 | 0 | 0 | | | |
| 12h | 13h | 32 | 71 | 0 | 0 | 71 | 9m23s | | 9m15s |
| 13h | 14h | 18 | 0 | 0 | 0 | 0 | | | |
| 15h | 16h | 31 | 0 | 0 | 0 | 0 | | | |
| 16h | 17h | 38 | 0 | 0 | 0 | 0 | | | |
| 17h | 18h | 2 | 89 | 0 | 0 | 89 | 22m47s | | 1h22m46s |

I.D.A. - Namur -

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Consolidation-dossier

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Consolidation-dossier
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 194 |
| Inception Number | 194 |
| Interruption Number | 22 |
| Resumption Number | 22 |
| Termination Number | 194 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 20m | 29m39s | 39m59s | 6m17s |
| Waiting Time | 0s | 9m43s | 28m43s | 8m36s |
| Intr Time by Intr | 0s | 0s | 0s | 0s |
| Intr Time by Process | | 0s | | |
| Idle Time by Incp/Rsmp | 0s | 6m7s | 1h | 30m53s |
| Idle Time by Process | | 6m48s | | |
| Elapse Time | | 45m10s | | |
| Numb of Proc in Wt State | 0.00 | 0.80 | 10.00 | 1.73 |
| Numb of Proc in Intr State | 0.00 | 0.00 | 7.00 | 0.00 |
| Numb of Proc in Actv State | 0.00 | 2.43 | 10.00 | 3.31 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Station-service-des-prets | 10.00 | 6.21 | | 6.21 | 1.27 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 11h | 12h | 107 | 85 | 0 | 0 | 45 | 5m38s | | 0s |
| 12h | 13h | 0 | 22 | 22 | 0 | 40 | 17m41s | | 0s |
| 13h | 14h | 0 | 0 | 0 | 22 | 22 | | 0s | 1h |
| 15h | 16h | 85 | 67 | 0 | 0 | 41 | 8m15s | | 0s |
| 16h | 17h | 2 | 20 | 0 | 0 | 44 | 23m9s | | 0s |
| 17h | 18h | 0 | 0 | 0 | 0 | 2 | | | |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Decision-credit

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Decision-credit
SPANS FROM 11h TO 12h, FROM 15h TO 16h ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 201 |
| Inception Number | 194 |
| Interruption Number | 8 |
| Resumption Number | 8 |
| Termination Number | 194 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 2m | 4m48s | 7m60s | 1m46s |
| Waiting Time | 0s | 24m22s | 1h21m42s | 22m45s |
| Intr Time by Intr | 0s | 0s | 0s | 0s |
| Intr Time by Process | | 0s | | |
| Idle Time by Incp/Rsmp | 0s | 5h26m49s | 19h11m39s | 8h17m9s |
| Idle Time by Process | | 5h40m17s | | |
| Elapse Time | | 5h57m37s | | |
| Numb of Proc in Wt State | 0.00 | 7.88 | 31.00 | 9.14 |
| Numb of Proc in Intr State | 0.00 | 0.00 | 2.00 | 0.00 |
| Numb of Proc in Actv State | 0.00 | 1.54 | 2.00 | 0.76 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Employe-service-credit | 7.00 | 4.38 | | 4.38 | 10.41 |
| Directeur-service-credit | 2.00 | 1.54 | | 1.54 | 7.88 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 08h | 09h | 12 | 0 | 0 | 0 | 0 | | | |
| 09h | 10h | 17 | 0 | 0 | 0 | 0 | | | |
| 10h | 11h | 25 | 0 | 0 | 0 | 0 | | | |
| 11h | 12h | 26 | 109 | 0 | 3 | 107 | 25m16s | 0s | 8h48m27s |
| 12h | 13h | 9 | 0 | 5 | 0 | 0 | | | |
| 13h | 14h | 5 | 0 | 0 | 0 | 0 | | | |
| 14h | 15h | 27 | 0 | 0 | 0 | 0 | | | |
| 15h | 16h | 37 | 85 | 0 | 5 | 87 | 23m13s | 0s | 1h15m53s |
| 16h | 17h | 30 | 0 | 3 | 0 | 0 | | | |
| 17h | 18h | 13 | 0 | 0 | 0 | 0 | | | |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Enreg-demande-pret

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Enreg-demande-pret
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 204 |
| Inception Number | 204 |
| Termination Number | 204 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 20m19s | 29m41s | 39m58s | 5m37s |
| Waiting Time | 0s | 2m47s | 32m52s | 6m59s |
| Elapse Time | | 32m27s | | |
| Numb of Proc in Wt State | 0.00 | 0.24 | 6.00 | 0.80 |
| Numb of Proc in Actv State | 0.00 | 2.55 | 8.00 | 1.76 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Station-service-des-prets | 10.00 | 6.21 | | 6.21 | 1.27 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 08h | 09h | 22 | 22 | 0 | 0 | 5 | 0s | | 0s |
| 09h | 10h | 28 | 28 | 0 | 0 | 30 | 0s | | 0s |
| 10h | 11h | 29 | 29 | 0 | 0 | 32 | 0s | | 0s |
| 11h | 12h | 10 | 10 | 0 | 0 | 21 | 2m56s | | 0s |
| 12h | 13h | 0 | 0 | 0 | 0 | 1 | | | |
| 13h | 14h | 27 | 27 | 0 | 0 | 4 | 0s | | 0s |
| 14h | 15h | 38 | 38 | 0 | 0 | 44 | 0s | | 0s |
| 15h | 16h | 31 | 23 | 0 | 0 | 29 | 7m35s | | 0s |
| 16h | 17h | 19 | 27 | 0 | 0 | 32 | 13m26s | | 0s |
| 17h | 18h | 0 | 0 | 0 | 0 | 6 | | | |

I.D.A. - Namur -

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Examen-credit

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Examen-credit
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 204 |
| Inception Number | 204 |
| Interruption Number | 48 |
| Resumption Number | 45 |
| Termination Number | 201 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 10m18s | 47m21s | 3h11m41s | 32m17s |
| Waiting Time | 0s | 4m27s | 43m24s | 8m32s |
| Intr Time by Intr | 0s | 1m12s | 16m42s | 2m57s |
| Intr Time by Process | | 0s | | |
| Idle Time by Incp/Rsmp | 0s | 57m50s | 15h | 3h6m3s |
| Idle Time by Process | | 1h10m34s | | |
| Elapse Time | | 2h1m36s | | |
| Numb of Proc in Wt State | 0.00 | 0.38 | 5.00 | 1.00 |
| Numb of Proc in Intr State | 0.00 | 0.02 | 7.00 | 0.17 |
| Numb of Proc in Actv State | 0.00 | .4.02 | 7.00 | 1.92 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Employe-service-credit | 7.00 | 4.38 | | 4.38 | 10.41 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 08h | 09h | 5 | 5 | 0 | 15 | 12 | 0s | 0s | 11h15m |
| 09h | 10h | 30 | 30 | 0 | 0 | 17 | 0s | | 0s |
| 10h | 11h | 32 | 29 | 0 | 0 | 25 | 16s | | 0s |
| 11h | 12h | 21 | 22 | 6 | 6 | 26 | 10m27s | 5m33s | 0s |
| 12h | 13h | 1 | 3 | 15 | 0 | 9 | 11m57s | | 0s |
| 13h | 14h | 4 | 4 | 0 | 15 | 5 | 0s | 0s | 47m22s |
| 14h | 15h | 44 | 41 | 0 | 0 | 27 | 1m12s | | 0s |
| 15h | 16h | 29 | 28 | 9 | 9 | 37 | 8m26s | 2m18s | 0s |
| 16h | 17h | 32 | 33 | 0 | 0 | 30 | 8m34s | | 0s |
| 17h | 18h | 6 | 9 | 18 | 0 | 13 | 7m31s | | 0s |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Preparation-dossier

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Preparation-dossier
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 204 |
| Inception Number | 204 |
| Interruption Number | 1 |
| Resumption Number | 1 |
| Termination Number | 204 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 10m2s | 15m | 19m59s | 2m56s |
| Waiting Time | 0s | 2m56s | 27m11s | 6m28s |
| Intr Time by Intr | 0s | 0s | 0s | 0s |
| Intr Time by Process | | 0s | | |
| Idle Time by Incp/Rsmp | 0s | 18s | 1h | 6m31s |
| Idle Time by Process | | 18s | | |
| Elapse Time | | 18m14s | | |
| Numb of Proc in Wt State | 0.00 | 0.25 | 4.00 | 0.63 |
| Numb of Proc in Intr State | 0.00 | 0.00 | 1.00 | 0.00 |
| Numb of Proc in Actv State | 0.00 | 1.28 | 7.00 | 1.25 |

| Required Resources : | Capacity Max | Capacity Used | Sharability Max | Sharability Used | Queue Size |
|---|---|---|---|---|---|
| Station-service-des-prets | 10.00 | 6.21 | | 6.21 | 1.27 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 08h | 09h | 5 | 5 | 0 | 0 | 0 | 0s | | 0s |
| 09h | 10h | 30 | 30 | 0 | 0 | 31 | 0s | | 0s |
| 10h | 11h | 32 | 32 | 0 | 0 | 23 | 0s | | 0s |
| 11h | 12h | 21 | 20 | 0 | 0 | 27 | 6m6s | | 0s |
| 12h | 13h | 1 | 2 | 1 | 0 | 7 | 11m57s | | 0s |
| 13h | 14h | 4 | 4 | 0 | 1 | 1 | 0s | 0s | 12m |
| 14h | 15h | 44 | 44 | 0 | 0 | 37 | 0s | | 0s |
| 15h | 16h | 29 | 25 | 0 | 0 | 28 | 8m38s | | 0s |
| 16h | 17h | 32 | 36 | 0 | 0 | 36 | 6m34s | | 0s |
| 17h | 18h | 6 | 6 | 0 | 0 | 14 | 0s | | 0s |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

PROCESS Refus-du-pret

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO PROCESS Refus-du-pret
SPANS FROM 12h TO 13h, FROM 17h TO 18h ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Triggering Number | 34 |
| Inception Number | 34 |
| Termination Number | 34 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Active Time | 2m | 2m | 2m | 0s |
| Waiting Time | 0s | 17m30s | 48m | 12m37s |
| Idle Time by Incp/Rsmp | 0s | 54m24s | 3h27m19s | 1h8m8s |
| Idle Time by Process | | 54m24s | | |
| Elapse Time | | 1h13m54s | | |
| Numb of Proc in Wt State | 0.00 | 1.01 | 11.00 | 1.93 |
| Numb of Proc in Actv State | 0.00 | 0.12 | 1.00 | 0.32 |

| Required Resources : | Capacity Max | Capacity Used | Shareability Max | Shareability Used | Queue Size |
|---|---|---|---|---|---|
| Imprimante-service-expedition | 1.00 | 0.66 | | 0.66 | 5.58 |

| [Low Bnd | Upp Bnd [ | Number of Trig | Incp | Intr | Resm | Term | Wait Time | Intr Time by Intr | Idle Time by Incp/Intr |
|---|---|---|---|---|---|---|---|---|---|
| 11h | 12h | 6 | 0 | 0 | 0 | 0 | | | |
| 12h | 13h | 8 | 14 | 0 | 0 | 14 | 10m4s | | 7m40s |
| 13h | 14h | 4 | 0 | 0 | 0 | 0 | | | |
| 15h | 16h | 10 | 0 | 0 | 0 | 0 | | | |
| 16h | 17h | 6 | 0 | 0 | 0 | 0 | | | |
| 17h | 18h | 0 | 20 | 0 | 0 | 20 | 22m42s | | 1h27m7s |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

SYNCHRONIZATION POINT Dossier-a-consolider

REPORTED BETWEEN 0d AND 5d

Realization Number 194

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Realization Time | 2m12s | 7h58m34s | 20h47m46s | 8h10m5s |
| Participation Time | | | | |
| TERM-DFDecision-credit | 0s | 11s | 14m37s | 1m25s |
| TERM-DFPreparation-dossier | 0s | 7h58m22s | 20h47m46s | 8h10m16s |

| Constraining Event | In Time | Par Freq |
|---|---|---|
| TERM-DFDecision-credit | 99.96% | 97.94% |
| TERM-DFPreparation-dossier | 0.04% | 2.06% |

I.D.A. - Namur -

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

RESOURCE Directeur-service-credit

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO RESOURCE Directeur-service-credit
SPANS FROM 11h TO 12h, FROM 15h TO 16h ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Maximum Capacity | 2.000 |
| Request Number | 209 |
| Allocate Number | 202 |
| Deallocate Number | 202 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Used Capacity | 0.0000 | 1.5427 | 2.0000 | 0.7596 |
| Used Snarability | 0.0000 | 1.5427 | 2.0000 | 0.7596 |
| Queue Size | 0.0000 | 7.8787 | 31.0000 | 9.1409 |

| PROCESS Which Requires this Resource | Wait Time | Intr Time by Intr | Actv Time | Numb Process in State Wait | Intr | Actv |
|---|---|---|---|---|---|---|
| Decision-credit | 24m22s | 0s | 4m48s | 7.879 | 0.000 | 1.543 |

| [Lower bound | Upper Bound[ | Number of Request | Alloc | Dealloc |
|---|---|---|---|---|
| 08h | 09h | 12 | 0 | 0 |
| 09h | 10h | 17 | 0 | 0 |
| 10h | 11h | 25 | 0 | 0 |
| 11h | 12h | 26 | 112 | 107 |
| 12h | 13h | 14 | 0 | 5 |
| 13h | 14h | 5 | 0 | 0 |
| 14h | 15h | 27 | 0 | 0 |
| 15h | 16h | 37 | 90 | 87 |
| 16h | 17h | 33 | 0 | 3 |
| 17h | 18h | 13 | 0 | 0 |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

RESOURCE Employe-service-credit

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO RESOURCE Employe-service-credit
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Maximum Capacity | 7.000 |
| Request Number | 461 |
| Allocate Number | 451 |
| Deallocate Number | 451 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Used Capacity | 0.0000 | 4.3801 | 7.0000 | 2.0853 |
| Used Sharability | 0.0000 | 4.3301 | 7.0000 | 2.0353 |
| Queue Size | 0.0000 | 10.4123 | 34.0000 | 8.6233 |

| PROCESS Which Requires this Resource | Wait Time | Intr Time by Intr | Actv Time | Numb Process in State Wait | Intr | Actv |
|---|---|---|---|---|---|---|
| Examen-credit | 4m27s | 1m12s | 47m21s | 0.379 | 0.023 | 4.017 |
| Decision-credit | 24m22s | 0s | 4m48s | 7.879 | 0.000 | 1.543 |

| [Lower bound | Upper Bound[ | Number of Request | Alloc | Dealloc |
|---|---|---|---|---|
| 08h | 09h | 17 | 20 | 12 |
| 09h | 10h | 47 | 30 | 17 |
| 10h | 11h | 57 | 29 | 25 |
| 11h | 12h | 53 | 140 | 139 |
| 12h | 13h | 30 | 3 | 29 |
| 13h | 14h | 9 | 19 | 5 |
| 14h | 15h | 71 | 41 | 27 |
| 15h | 16h | 76 | 127 | 133 |
| 16h | 17h | 65 | 33 | 33 |
| 17h | 18h | 37 | 9 | 31 |

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

RESOURCE Imprimante-service-expedition

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO RESOURCE Imprimante-service-expedition
SPANS FROM 12h TO 13h, FROM 17h TO 18h ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Maximum Capacity | 1.000 |
| Request Number | 194 |
| Allocate Number | 194 |
| Deallocate Number | 194 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Used Capacity | 0.0000 | 0.6576 | 1.0000 | 0.4745 |
| Used Sharability | 0.0000 | 0.6576 | 1.0000 | 0.4745 |
| Queue Size | 0.0000 | 5.5760 | 27.0000 | 6.8815 |

| PROCESS Which Requires this Resource | Wait Time | Intr Time by Intr | Actv Time | Numb Process in State Wait | Intr | Actv |
|---|---|---|---|---|---|---|
| Acceptation-du-pret | 16m51s | 0s | 2m | 4.568 | 0.000 | 0.542 |
| Refus-du-pret | 17m30s | 0s | 2m | 1.012 | 0.000 | 0.116 |

| [Lower bound | Upper Bound[ | Number of Request | Alloc | Dealloc |
|---|---|---|---|---|
| 11h | 12h | 45 | 0 | 0 |
| 12h | 13h | 40 | 85 | 85 |
| 13h | 14h | 22 | 0 | 0 |
| 15h | 16h | 41 | 0 | 0 |
| 16h | 17h | 44 | 0 | 0 |
| 17h | 18h | 2 | 109 | 109 |

I.D.A. - Namur -

Global Statistics :GESTION DE PRETS - 1 demande toutes les (+/-) 12 minutes

RESOURCE Station-service-des-prets

REPORTED BETWEEN 0d AND 5d

DEFINE CALENDAR RELATED TO RESOURCE Station-service-des-prets
SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

| | |
|---|---|
| Maximum Capacity | 10.000 |
| Request Number | 625 |
| Allocate Number | 625 |
| Deallocate Number | 625 |

| | Min | Mean | Max | Std Dev |
|---|---|---|---|---|
| Used Capacity | 0.0000 | 6.2117 | 10.0000 | 3.0871 |
| Used Sharability | 0.0000 | 6.2117 | 10.0000 | 3.0871 |
| Queue Size | 0.0000 | 1.2735 | 13.0000 | 2.6445 |

| PROCESS Which Requires this Resource | Wait Time | Intr Time by Intr | Actv Time | Numb Process in State Wait | Intr | Actv |
|---|---|---|---|---|---|---|
| Enreg-demande-pret | 2m47s | 0s | 29m41s | 0.238 | 0.000 | 2.547 |
| Preparation-dossier | 2m56s | 0s | 15m | 0.250 | 0.000 | 1.279 |
| Consolidation-dossier | 9m43s | 0s | 29m39s | 0.797 | 0.000 | 2.433 |

| [Lower bound | Upper Bound[ | Number of Request | Alloc | Dealloc |
|---|---|---|---|---|
| 08h | 09h | 27 | 27 | 5 |
| 09h | 10h | 58 | 58 | 61 |
| 10h | 11h | 61 | 61 | 55 |
| 11h | 12h | 138 | 115 | 93 |
| 12h | 13h | 24 | 24 | 71 |
| 13h | 14h | 31 | 54 | 27 |
| 14h | 15h | 82 | 82 | 81 |
| 15h | 16h | 145 | 115 | 98 |
| 16h | 17h | 53 | 83 | 112 |
| 17h | 18h | 6 | 6 | 22 |