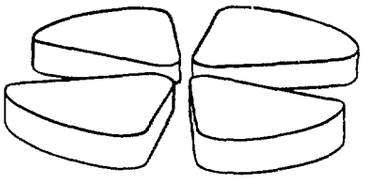
GANIL



Nouveau système de contrôle

SUPERVISION

IMAGIN EN ADA

GANIL A 91 06

Secteur Exploitation
Informatique Machine
C. MAUGEAIS

GANIL 503/91
le 08/01/91

Nouveau système de contrôle
SUPERVISION
IMAGIN EN ADA

Tous les nouveaux applicatifs pour IMAGIN sont désormais écrits en ADA. Pour maintenir une structure cohérente entre les différents programmes utilisateurs, il a été mis au point une ossature utilisable dans toutes les applications.

Les différents fichiers qui constituent la partie applicatif du programme sont directement copiés lors de la création d'une nouvelle application dans le répertoire associé à cet applicatif chez l'utilisateur. Le programme principal porte le nom de l'application et la librairie ACS associée porte le nom de l'application avec le suffixe _ACS.

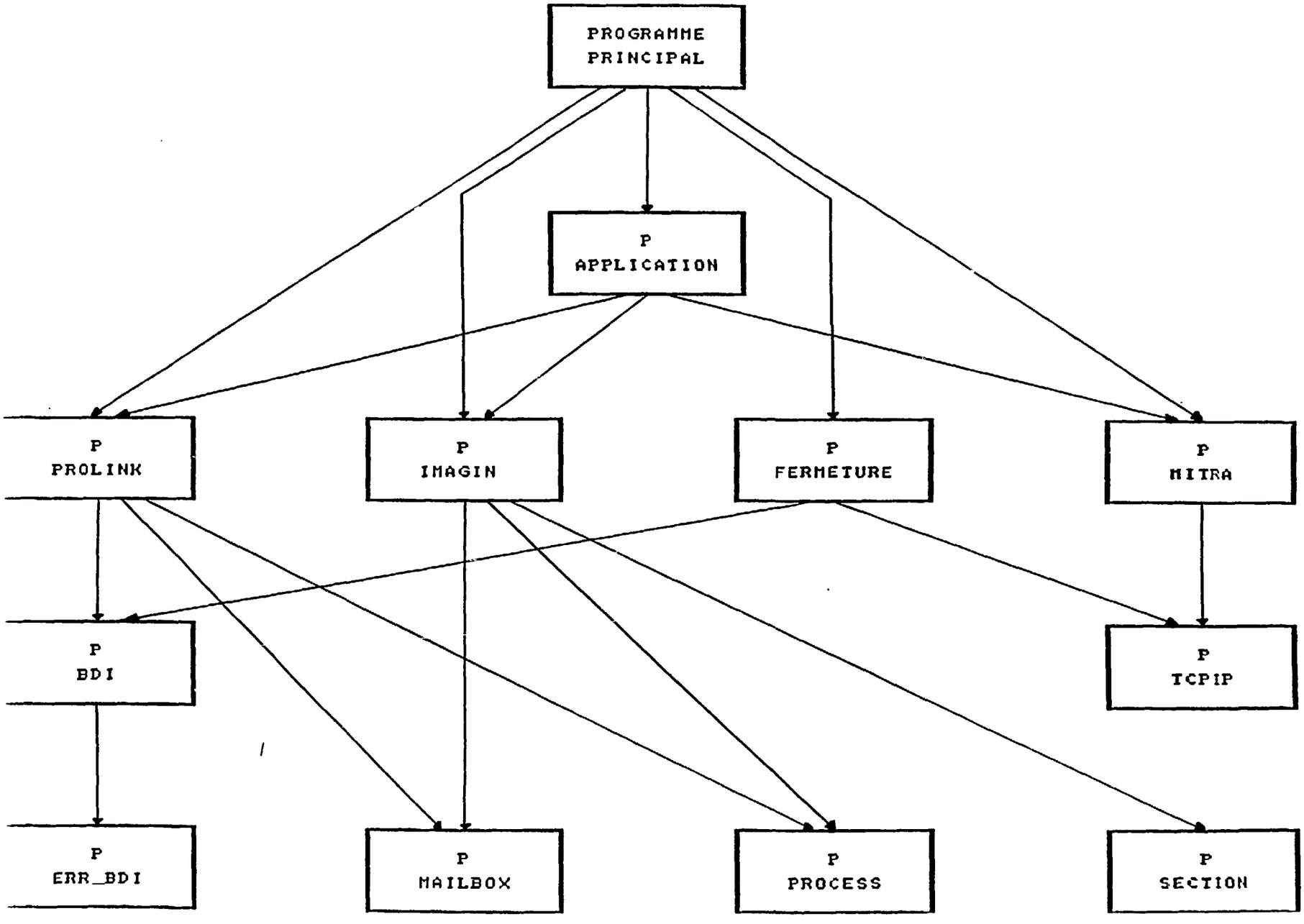
Pour éviter à l'utilisateur de traiter directement les dialogues avec l'animateur IMAGIN, cette partie seulement de la structure est visible et modifiable alors que la partie cachée prend en charge les initialisations et les interfaces système.

I La structuration

1) La structure par packages

Mise à part la procédure principale de l'application, les différentes procédures du programme sont inclues dans des packages. Cela permet d'avoir des unités de compilation pas trop importantes et de pouvoir déclarer des constantes ou des types visibles dans plusieurs procédures.

Les fonctions de création sont placées dans les différentes parties élaboratives des packages P_IMAGIN, P_MITRA ou P_PROLINK. La partie élaborative d'un package se situe après le 'begin' en fin de corps du package et elle est exécutée avant le début du programme applicatif


PROGRAMME
PRINCIPAL
P
APPLICATION
P
PROLINK
P
IMAGIN
P
FERMETURE
P
MITRA
P
BDI
P
TCPIP
P
ERR_BDI
P
MAILBOX
P
PROCESS
P
SECTION

2) Les différentes procédures

Le dialogue animateur applicatif se fait par l'intermédiaire de différentes fonctions (Demande de rafraîchissement, Set bit, Reset bit...).

Pour chacune de ces fonctions, il existe une procédure appelée par le programme principal et qui sera remplie par l'utilisateur suivant ses besoins. Il a également été crée quelques procédures supplémentaires (ex TERMINER permet à l'utilisateur de fermer proprement les ressources externes mises en oeuvre dans son programme).

Afin de ne pas surcharger les fichiers, l'utilisateur pourra créer dans des fichiers distincts des procédures en mode "separate" qui seront appelées par les procédures préexistantes. Ceci n'est intéressant que pour des procédures relativement volumineuses.

La procédure qui porte le nom de l'application est la procédure principale. Elle assure la communication avec l'animateur, contient une boucle sur condition d'arrêt et appelle en fin de traitement la procédure FERMER issue du package P_FERMETURE.

3) Les différents packages

Le package P_APPLICATION contient des procédures vides appelées à partir de la procédure principale suivant le type de message reçu de l'animateur. Il permet aussi de déclarer des variables visibles de toutes les procédures du package.

Les autres packages contiennent les fonctions et procédures que l'utilisateur peut appeler :

Le package P_PROLINK permet le dialogue entre Prolink + et l'applicatif, en proposant les procédures d'envoi de messages et de traitement des fiches de la base de données industrielle.

Le package P_IMAGIN permet le dialogue entre Imagin et l'applicatif, en proposant les procédures d'envoi de messages et de traitement des erreurs.

Le package P_MITRA permet le dialogue avec le Mitra, en proposant des procédures de demande et de réception des données ainsi que de fermeture propre du dialogue.

Le package P_FERMETURE permet de libérer proprement toutes les ressources allouées à l'applicatif dans les différents packages cachés.

II Les initialisations

1) Les boites aux lettres

Le dialogue avec l'animateur exige la création de trois boites aux lettres, une recevant les messages (RECEP), une pour les accusés de réception (ACKN), et une pour les messages envoyés par l'applicatif (ENVOI).

Les fonctions de création sont placées dans la partie élaboration du package P_IMAGIN où sont utilisées les boites. Une initialisation identique est mise en oeuvre pour le dialogue avec Prolink +, elle se situe dans le package P_PROLINK.

2) La section globale

Il s'agit d'une zone mémoire commune à l'animateur et à l'applicatif par où sont transférées les données. Le mapping sur cette zone se fait dans P_APPLICATION par appel de la fonction MAPPER_TABLE qui est dans PACK_IMAGIN.

Dans P_APPLICATION, on déclare autant de records représentant les données à transmettre à l'animateur que le synoptique comporte de vues, chaque record étant placé physiquement au même endroit. On pourra ainsi écrire indifféremment dans l'un ou l'autre des records. Ces records ne doivent jamais comporter de valeurs par défaut.

3) *Le lancement de l'animateur*

Le lancement de l'animateur se fait dans l'élaboration du package P_IMAGIN.

Grâce à l'option PRC_M_DETACH le process lancé est indépendant du process père (l'applicatif). Tous les paramètres de configuration du lancement sont établis à l'avance et se réalisent donc de façon transparente.

4) Le chaînage père fils

Pour éviter, à l'utilisation, des temps de chargement trop longs à chaque changement de synoptique, tous les synoptiques IMAGIN ont été regroupés dans un seul. Celui-ci est animé par un applicatif père qui lance, lors d'un changement de "sous-synoptique", l'applicatif fils correspondant.

Tout applicatif doit donc pouvoir fonctionner en fils ou en orphelin. C'est pourquoi on teste s'il est orphelin ou non dans l'élaboration du package P_IMAGIN et on réalise selon les cas les initialisations nécessaires. On peut connaître le résultat dans toute procédure grâce à la fonction ETRE_ORPHELIN du package P_IMAGIN.

III Le fonctionnement

1) La procédure principale

La procédure principale est construite sur une boucle sans fin dont la condition de sortie est la fin du synoptique ou une erreur.

Elle commence par se mettre en attente d'un message de l'animateur. Quand ce message arrive, un acknowledge est renvoyé (le numéro de fonction suivie de 0 si la fonction demandée est correcte et de 1 sinon)

Ensuite, la procédure correspondant à la fonction demandée est appelée. Si le message est un message de fin, la variable booléenne FIN_SYN devient vraie et on ne passera plus dans la boucle.

Dans le cas d'une demande de rafraîchissement, il y a un traitement spécial. Le message reçu lors de cette demande contient comme information le nom de la vue courante. On va donc tester ce nom pour savoir quelle procédure appeler. Si l'applicatif est orphelin, tout est simple. Il suffit de remplir la section globale et de répondre à l'animateur par la procédure DEMANDER_TABLE.

Par contre, si l'applicatif est fils, il existe trois configurations possibles lors de la réception du message DEMTAB. L'applicatif fils risque de démarrer avant que la vue TRONC (vue principale du synoptique général) n'ait été remplacée par la vue principale du "sous-synoptique". C'est pourquoi on teste cette condition avant l'appel de DEMANDER_TABLE et on appelle alors DEMANDER_TRONC, qui ne fait aucun remplissage de la section globale.

En cours de traitement, le nom de vue correspond au nom de l'une des vues du synoptique fils, on met donc à jour la zone globale selon la structure appropriée et on répond à l'animateur par la procédure DEMANDER_TABLE.

L'applicatif fils devra se terminer s'il y a retour à la vue principale. Le message reçu lors de cette demande contient alors le nom de la vue TRONC. Si c'est le cas, l'applicatif fils se termine de lui même en appelant la procédure TERMINER qui permet de libérer proprement les ressources et en sortant de la boucle de traitement.

Ce programme principal ne nécessite en principe aucune modification pour être utilisé dans une application réelle.

2) Les procédures de traitement

Toutes les procédures sont écrites à l'avance mais la plupart sont vides. L'utilisateur peut ne pas avoir besoin de modifier tous les procédures, **IL NE FAUT SURTOUT PAS** supprimer les procédures qui restent inchangées car il y aurait alors incohérence avec le programme principal.

Les procédures qui sont développées ne le sont qu'à titre d'exemple. Il faut les reprendre pour les adapter à l'application en cours.

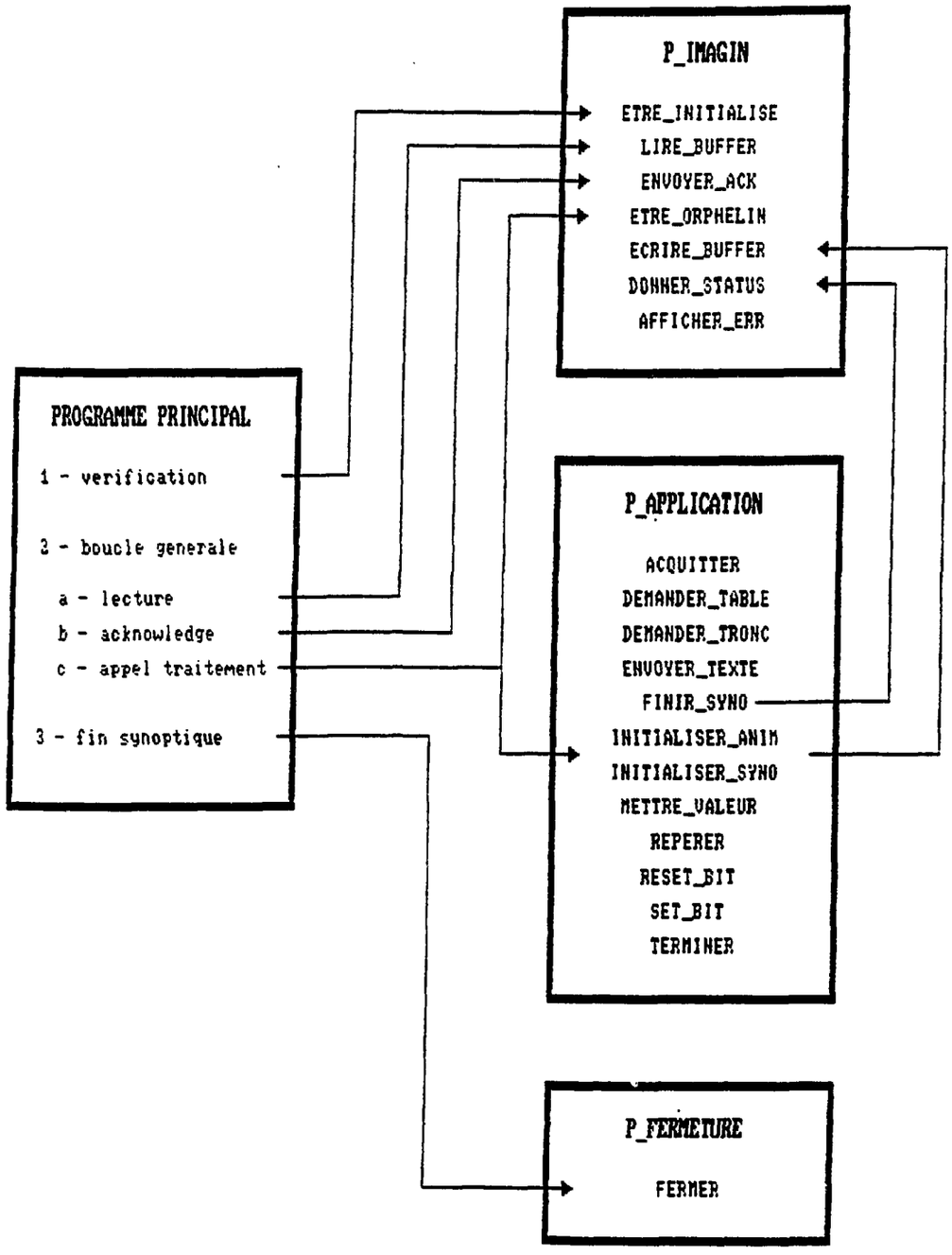
IV Les différents messages

1) Les messages reçus

Il n'existe qu'un type (le type MESSAGE) pour tous les messages reçus mais c'est un article à parties variantes c'est à dire que suivant la fonction demandée par l'animateur le type est différent. Ce type est décrit dans les spécifications de P_APPLICATION.

Le premier élément de ce type est donc le numéro de fonction du message. La variable R_BUF_REC dans laquelle sont reçus les messages est déclarée de type MESSAGE sans affectation précise mais dès l'utilisation elle se trouve bloquée dans un type jusqu'à la prochaine arrivée de message.

Lors d'un envoi de message de l'animateur vers l'applicatif, le buffer est automatiquement transféré à la procédure correspondante. Cette variable contient les paramètres nécessaires à l'exécution de la fonction demandée par l'animateur.


P_IMAGIN
ETRE_INITIALISE
LIRE_BUFFER
ENVOYER_ACK
ETRE_ORPHELIN
ECRIRE_BUFFER
DONNER_STATUS
AFFICHER_ERR
PROGRAMME PRINCIPAL
1 - verification
2 - boucle generale
a - lecture
b - acknowledge
c - appel traitement
3 - fin synoptique
P_APPLICATION
ACQUITTER
DEMANDER_TABLE
DEMANDER_TRONC
ENVOYER_TEXTE
FINIR_SYNO
INITIALISER_ANIM
INITIALISER_SYNO
METTRE_VALEUR
REPERER
RESET_BIT
SET_BIT
TERMINER
P_FERMETURE
FERMER

2) Les messages d'accusé de réception

Il existe deux types différents pour l'accusé de réception. On choisit au début de la procédure principale si on souhaite accepter ou non certains messages issus de l'animateur. Pour accepter un message, il suffit de positionner à true l'élément correspondant du tableau FONCTION_ACQUITTEE. L'accusé de réception se fera ensuite sur simple appel de la procédure ENVOYER_ACK issue du package P_IMAGIN.

3) Les messages d'envoi

Il n'existe qu'un type (le type REPONSE) pour tous les messages reçus mais c'est un article à parties variantes c'est à dire que suivant la fonction demandée par l'animateur le type utilisé est différent. Ce type est décrit dans le body de P_APPLICATION.

Le premier élément de ce type est donc le numéro de fonction du message. La variable R_BUF_EMIS par laquelle sont envoyés les messages est déclarée de type REPONSE avec l'affectation précise correspondant au traitement désiré. Elle se trouve bloquée dans ce type de façon définitive.

De plus il existe dans le package P_IMAGIN une procédure spéciale, ECRIRE_BUFFER, qui est instanciée pour le type variant. Ainsi l'envoi de message est transparent pour l'utilisateur. La procédure instanciée ECRIRE_REPONSE envoie directement le message dans la boite aux lettres appropriée.

Exemple : pour envoyer le message DEMTAB_REP il suffit de déclarer une variable de ce type : R_BUF_EMIS : REPONSE (NO_FONCT => NO_DEMTAB_REP), de remplir correctement les différents champs et d'utiliser ensuite la procédure de façon classique : ECRIRE_REPONSE (REPONSE => R_BUF_EMIS)

3) Rafraîchissement de l'animateur :

Celui-ci se fait par écriture dans une zone mémoire commune à l'applicatif et à l'animateur. C'est après chaque demande de rafraîchissement (DEMTAB) que l'animateur vient lire la zone commune.

Cette zone doit être remplie par un type record contenant exactement toutes les variables nécessaires au rafraîchissement. Il est plus simple d'associer un record par vue du synoptique. Ce record permet de ne mapper que la taille mémoire exacte nécessaire à l'application.

Les différents records possibles sont mappés sur le même espace mémoire, l'utilisateur peut donc écrire indifféremment dans l'un ou l'autre des records, en fonction de la vue courante qu'il traite.

V Le traitement des erreurs

1) Le but

Le traitement des erreurs a pour but de ne pas stopper brutalement le programme en cas d'erreur. De plus, il permet de garder une trace en stockant les messages d'erreur dans un fichier. Enfin, il est possible ainsi d'afficher à l'écran de surveillance ses propres messages d'erreur.

2) *Les types d'erreurs*

Dans l'applicatif, nous avons surtout traité les erreurs VMS. Ces erreurs apparaissent dans les packages d'interface avec le système et sont traitées de façon transparente. Elles sont stockées dans un fichier err_appli.err, entièrement géré par les packages d'interface système.

Toutefois, les erreurs ADA des packages utilisateur remontent, sont affichées par le programme principal et provoquent l'arrêt du synoptique si elles ne sont pas traitées de façon appropriées par l'utilisateur.

De plus, il existe une procédure DONNER_STATUS qui détermine la cause de l'arrêt de l'animateur grâce au STATUS envoyé dans le message de fin. S'il y a eu erreur, le message correspondant est envoyé dans le fichier err_appli.err.

3) La surveillance

La surveillance est un programme fourni par Sferca qui permet d'afficher sur un écran réservé tous les messages envoyés à ce programme. Cela permet d'afficher les divers messages d'erreur survenus lors de l'exécution des programmes applicatifs.

La procédure d'envoi de messages à la surveillance s'appelle AFFICHER_ERR et se trouve aussi dans P_IMAGIN.

Dans l'hypothèse où quelque chose a été envoyé à la surveillance, le message s'affiche sur l'écran spécialisé et est également sauvegardé dans un fichier géré par la surveillance.

4) La remontée des erreurs

Il n'est pas possible d'arrêter le programme pendant les élaborations. C'est pourquoi il a été nécessaire de transmettre l'information "il n'y a pas eu erreur" par la fonction ETRE_INITIALISE du package P_IMAGIN pour Imagin ou ETRE_INITIALISE du package P_PROLINK pour Prolink +. De cette manière, s'il y a eu erreur, le programme ne démarre pas.

Il y a levée et remontée d'exception dans toutes les procédures du programme. Les exceptions passent automatiquement à travers les procédures utilisateur dans la clause " WHEN OTHERS " qui sera à utiliser dans le traitement uniquement avec l'instruction RAISE.

Lorsqu'une erreur remonte ainsi, il est nécessaire qu'elle atteigne le programme principal qui peut alors se terminer proprement en fermant toutes les ressources utilisées dans le programme. Les ressources système sont fermées dans la procédure FERMER du package P_FERMETURE, les ressources utilisateur dans la procédure TERMINER du package P_APPLICATION.

L'utilisateur peut donc faire un traitement de ses propres erreurs dans le traitement des exceptions et des ses propres ressources dans le procédure TERMINER.