IASI

THALES INFORMATION SYSTEMS

IA-DLR-2100-9546-THA
Edition : 02 Date : 23/08/2002
Révision : 02 Date : 24/10/2002
Réf. : - MT : X Code diffusion : E

DOSSIER DES LOGICIELS REUTILISES
DOSSIER DES LOGICIELS RÉUTILISÉS POUR L'OPS IASI

Rédigé par :		le : 24-10-02	
CABANE Philippe	THALES IS		
PASCAL Jean-Luc	THALES IS		
Validé par :		le : 24/10/02	
BRANET Pascal	THALES IS		
AYER Patrick	THALES IS		
Pour application :		le :	
MORENO Richard	DTS/MID/VM/TD		

18, avenue Edouard Belin 31401 Toulouse CEDEX 4 Tél. 05 61 27 31 31

BORDEREAU D'INDEXATION

CONFIDENTIALITE : **NC**	**MOTS CLES : Conception, Réutilisation, méthodologie, IASI, Traitement d'images**

TITRE DU DOCUMENT : Dossier des Logiciels Réutilisés

Dossier des Logiciels Réutilisés pour l'OPS IASI

AUTEUR(S) : CABANE Philippe THALES IS
PASCAL Jean-Luc THALES IS

RESUME : Décrit les logiciels réutilisés dans le cadre de l'OPS IASI. Ce document a pour but de présenter notre analyse qui a conduit à identifier le niveau de réutilisation envisagé pour l'OPS.

DOCUMENTS RATTACHES : Ce document vit seul.	**LOCALISATION :**

VOLUME : 1	**NBRE TOTAL DE PAGES : 45** **DONT PAGES LIMINAIRES : 7** **NBRE DE PAGES SUPPL. : 0**	**DOCUMENT COMPOSITE : N**	**LANGUE : FR**

GESTION DE CONF. : F	**RESP. GEST. CONF. : GOMEZ MH**

CAUSE D'EVOLUTION : Correction suite aux FEPS n° 10 et 29 du PKCD.

CONTRAT : 01/8937

SYSTEME HOTE : Microsoft Word 8.0b, \\NWTLS10\PROJETS\PROJETS\IASI\Modèles CNES\CnesIndustriel97.dot v2.0.4

DIFFUSION INTERNE

Nom	Sigle	BPi	Observations
BLUMSTEIN Denis	DSO/OT/SE/IA	2504	
CHALON Gilles	DSO/OT/SE/IA	2504	
PONCE Ghislaine	DSO/OT/SE/IA	2504	
SEGALEN Barbara	DSO/SG/CS	2504	
DUPLAA Michel	DTS/MID/VM/D	1502	
MARQUIER Henry	DTS/MID/VM/TD	1502	
MORENO Richard	DTS/MID/VM/TD	1502	
GOMEZ Marie-Hélène	DTS/MID/VM/MG	1502	
BAILLY Isabelle	DSO/OT/QTIS/VP	811	
RAYSSIGUIER Michel	DTS/OT/QTIS/VP	811	
MATHIEU Nathalie	DTS/AQ/QIS/SC	1415	
RICHARD Pascal	DEE/IR/ISM/IS	1311	

DIFFUSION EXTERNE

Nom	Sigle	Observations
AYER Patrick	THALES IS	
BOBIN Serge	THALES IS	
BRANET Pascal	THALES IS	
PASCAL Jean-Luc	THALES IS	
CABANE Philippe	THALES IS	

MODIFICATION

Ed.	Rév.	Date	Référence, Auteur(s), Causes d'évolution
02	02	24/10/2002	- CABANE Philippe THALES IS PASCAL Jean-Luc THALES IS Correction suite aux FEPS n° 10 et 29 du PKCD.
02	01	18/09/2002	- Correction suite aux FEPS de la RCP : NP02, NP03 CABANE Philippe THALES IS PASCAL Jean-Luc THALES IS
02	00	23/08/2002	- CABANE Philippe THALES IS PASCAL Jean-Luc THALES IS Mise au format GDOC
01	00	14/06/2002	- CABANE Philippe THALES IS PASCAL Jean-Luc THALES IS Création du document

SOMMAIRE

GLOSSAIRE ET LISTE DES PARAMETRES AC & AD

API	Applicative Program Interface
CGS	Core Ground Segment : segment-sol développé par ALCATEL sous contrat d'EUMETSAT, et dans lequel l'OPS ira s'insérer
GDD	Gestionnaire de Données et de Diffusion : sous-système du CNES
IASI	Infrared Atmospheric Sounding Interferometer : interferomètre de sondage atmosphériq'ue dans l'infrarouge. L'objet de ce document est de spécifier le logiciel opérationnel du traitement sol des données IASI.
IHM	Interface Homme Machine
MCS	Monitoring and Control Segment
MLA	MCS Local Agent
OPS	Logiciel Opérationnel (Operational Software) : correspond au IASI level 1 PPS dans les glossaires d'EUMETSAT. PPS=Product Processing Software
PDS	Payload Data Segment : Centre de Mission ENVISAT
PGE	Product Generation Element : fournit des services aux PPS [DA1]
PGF	Product Generation Facility
SIF	Simulteurs d'Interfaces du F-PAC ENVISAT (sous système du CNES)
SIP	Sub Instruction Performer : process de l'architecture du SIF
TEC	Technical Expertise Center : CET en anglais
UML	Unified Modelling Language
WO	Working Order

Liste des paramètres AC :

Liste des paramètres AD :

1.GENERALITES

1.1.DOCUMENTS DE REFERENCE ET APPLICABLES

La liste des documents constituant le référentiel du projet OPS-IASI est détaillée dans la « Liste Unique » du logiciel OPS-IASI [DR100].

2.INTRODUCTION

2.1.OBJECTIF

Ce document présente les logiciels réutilisés dans le cadre du développement du logiciel OPS IASI (chaîne de niveau 1), ainsi que les conditions requises pour leur intégration.

Les logiciels concernés sont :

- **le Serveur de données SSALTO** (**S**egment **S**ol multi-missions **ALT**imétrie, **O**rbitographie et localisation précise) dont la fonction est d'assurer les échanges de données entre différentes entités des filières DORIS, POSEIDON,
- **SIF** (**S**imulateur d'**I**nterface du **F**rench PAC) qui implémente l'interface entre le centre de mission ENVISAT et le système de production de données GDD.

Ces deux logiciels sont la propriété du CNES.

Pour chacun des logiciels identifiés, les éléments suivants sont présentés dans les chapitres qui suivent :

- Présentation du logiciel existant :
  - description des fonctions le constituant,
  - présentation de la documentation associée au logiciel,
  - contexte de développement.
- Analyse de son adéquation aux besoins :
  - adéquation aux besoins techniques,
  - adéquation aux exigences qualité.
- Analyse fine des composants :
  - le rôle du composant dans la nouvelle architecture de l'OPS IASI,
  - le niveau de réutilisation envisagé ainsi qu'une première analyse des modifications à apporter à l'existant,
  - une conclusion sur la réutilisation *à priori* et sur les efforts associés.

Ce document doit être vu comme l'analyse préalable et indispensable qui nous a permis d'aborder la conception préliminaire de l'OPS. C'est à partir de celle-ci que nous avons construit une solution pertinente. Cette solution est détaillée dans le dossier de Conception Préliminaire [DA108].

3.REUTILISATION DU LOGICIEL SSALTO

3.1.PRESENTATION DU LOGICIEL

3.1.1.Fonctionnalités

Le Segment Sol multi-missions SSALTO regroupe l'ensemble des moyens sol nécessaires pour assurer :

- le contrôle des instruments embarqués de la filière DORIS et de la filière POSEIDON, le contrôle du réseau de balises sol DORIS,
- le traitement des données des charges utiles des missions DORIS/SPOT et des missions altimétriques TOPEX/POSEIDON, JASON-1 et ENVISAT.

Le serveur de données a en charge d'assurer les échanges de données entre les différentes entités internes et externes à SSALTO. Ces échanges consistent à importer ou exporter des fichiers de manière sécurisée via un serveur agréé. Ce serveur gère tous les échanges, vis-à-vis de l'extérieur, de SSALTO.

3.1.2.Conditions de Réalisation

3.1.2.1.Développement du Produit

Le serveur de donnée(SD) SSALTO a été réalisé par THALES IS en 1999 pour le compte du CNES.

Le logiciel comporte quatre composants : SD, SA, ADM et CMN. La structure du produit est décrite dans le document de conception préliminaire de SSALTO.

Elle est rappelée ci-dessous pour mémoire :

Organigramme produit SSALTO

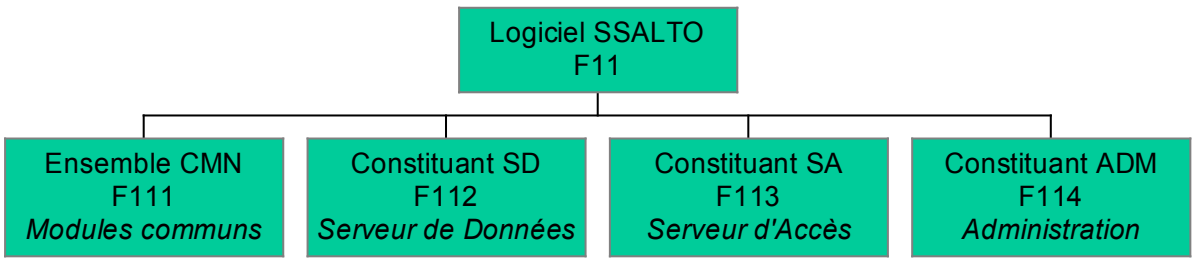

Figure 1 : Organigramme Produit SSALTO

La démarche de développement adoptée pour la réalisation de ce produit logiciel a consisté en un cycle de développement en V.

3.1.2.2.Condition de Garantie et de Maintenance

Le serveur de données SSALTO est actuellement maintenu sur le site du CNES Toulouse par THALES IS.

La dernière version de SSALTO est la V2.4.

3.1.3.Documentation Associée

Le projet SD de SSALTO a été réalisé sur la base des exigences applicables aux marchés de classe 1. La documentation projet est ainsi claire et suffisante, elle est résumée dans le tableau ci-dessous :

Référence	Intitulé
SMM-ST-M4-EA-20304-SYS	Spécification Logicielle de SSALTO
SMM-CP-M4-EA-20315-SYS	Conception Préliminaire de SSALTO.
SMM-DD-M4-EA-20324-SYS	Dossier de définition - Composant SD
SMM-DD-M4-EA-20323-SYS	Dossier de définition - Composant SA
SMM-DD-M4-EA-20322-SYS	Dossier de définition - Composant ADM
SMM-DD-M4-EA-20325-SYS	Dossier de définition - Composant CMN
SMM-PG-M4-EA-20302-SYS	Plan d'application de SSALTO
SMM-M4-EA-20310-SYS	Standard de codage
SMM-BQ-M4-EA-20313-SYS	Bilan qualité
SMM-PG-M4-EA-20302-SYS	Plan d'Application
SMM-PG-M4-EA-20303-SYS	Plan de Gestion de Configuration
SMM-PG-M4-EA-20306-SYS	Plan d'Assurance Sécurité
SMM-PE-M4-EA-20307-SYS	Plan d'Essai d'intégration et de validation
SMM-MU-M4-EA-20312-SYS	Manuel d'utilisation
SMM-PR-M4-EA-20311-SYS	Cahier de Recette du Logiciel

Tableau 1 - Documentation du Projet SSALTO

3.1.4.Environnement Matériel et Logiciel

Le logiciel SD de SSALTO a été développé avec une méthodologie objet en utilisant les moyens suivants :

- Spécifications, Conception : UML, Rational Rose,
- Langage :C, C++,
- Développement : PC sous LINUX et WINDOWS 95,
- Machine cible : SUN sous Solaris 2.7.

3.1.5.Architecture logicielle

3.1.5.1.Présentation

Le logiciel SD de SSALTO s'articule autour de quatre composants : SD, SA, ADM et CMN.

- le composant **SA** ou Serveur d'Accès est un serveur de données FTP agrée.
- les deux composants SD et ADM constituent le cœur de l'application :
  - le composant **SD** ou Serveur de Données est en charge de collecter et diffuser périodiquement ou à la demande des fichiers de donnée à travers le SA,
  - le composant **ADM** est l'outil d'administration du SD-SSALTO qui permet à l'administrateur de configurer, contrôler et suivre les travaux du sous-système.
- le composant **CMN** est la librairie dans modules communs.

La figure suivante présente l'architecture logicielle et le déploiement des composants de SSALTO :

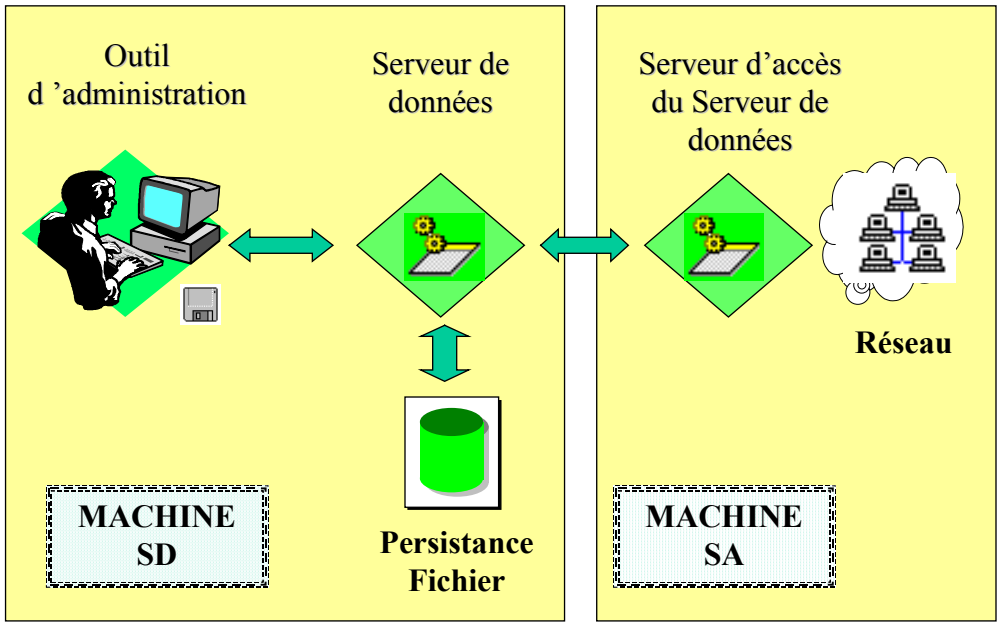

Figure 2 : Déploiement des Composants du SD-SSALTO

3.1.5.2.Caractéristiques Principales des Composants

Le composant ADM

Le composant ADM est constitué d'un seul processus et est monothread. Le processus ADM est lancé sur une commande d'un utilisateur pour le login spécifique à l'application Serveur de données SSALTO. Ce processus s'arrête lorsque l'utilisateur quitte l'application.

Le composant SD

Le composant SD est constitué d'un processus multi-threadé (POSIX) qui gère un échéancier (matérialisé par un fichier) regroupant toutes les tâches à effectuer par l'application.

Le composant SA

Le composant SA est constitué de 3 processus. Tous sont monothread. Le processus de supervision (Superviseur) lance, à la demande du SD, les processus Importateur ou Exportateur. Le Superviseur lance 1 processus par commande d'importation ou d'exportation. Le Superviseur est lancé à la fin de la phase de boot du système. Il ne s'arrête jamais. Les processus Importateur ou Exportateur s'arrêtent d'eux même à la fin du transfert de fichier (réussi ou non). Le Superviseur peut aussi, à la demande du SD, arrêter l'exécution d'un processus importateur ou exportateur.

3.2.INTERET DE LA REUTILISATION

3.2.1.Adéquation aux besoins techniques

Nous avons besoin, dans le cadre du lancement des traitements algorithmiques, d'un processus multi-threadé. Afin d'être évolutive et paramétrable, une architecture multithreads doit respecter une organisation et des principes de construction. THALES IS a réalisé pour le compte du CNES en 1999 le serveur de données SSALTO qui implémente une telle architecture : un processus multi-threadé qui gère un échéancier de tâches. Les besoins et les fonctionnalités entre l'OPS et SSALTO étant très similaires, nous proposons de récupérer l'architecture SSALTO de gestion de tâches parallèles à l'aide de threads. Ainsi, nous proposons de réutiliser les composants CMN et SD du logiciel SSALTO.

L'intérêt de la réutilisation de cette architecture multi-threadée dans le cadre de l'OPS IASI semble ainsi évident ; il nous permettra de bénéficier rapidement d'une solution souple et fiable.

Les points marquants sont :

- le composant SD constitue la souche du serveur de traitements de l'OPS son architecture est conservée,
- l'échéancier du serveur SD est revu et corrigé pour prendre en compte les spécificités des tâches de l'OPS qui sont des enchaînements de traitements algorithmiques,
- le composant CMN est conservé en effet il regroupe les services communs, entre autres, utilisés par le SD et l'échéancier,
- les composants ADM et SA sont abandonnés.

3.2.2.Adéquation aux besoins qualités

Le logiciel SD de SSALTO a été développé en respectant des règles strictes de qualité répertoriées dans son Plan d'Application. Ces besoins sont très proches des contraintes imposées sur le système IASI OPS dans le cadre des marchés de classe 1. En particulier les méthodes et outils utilisés dans les différentes phases de développement sont résumés ci dessous :

PHASE	METHODE	OUTIL
Conception préliminaire	méthode orientée objet notation UML	Rational Rose
Conception détaillée	Raffinement des classes Entête de classe et de méthode + Pseudo code	Outil d'extraction de la CD à partir des sources.
Codage	MPM C++ Normes de codage C++	C++ Logiscope statique (C++ Code checker)
Tests Unitaires	Stratégie de test unitaire décrite dans les Dossiers de Définition des composants.	
Intégration/validation sous-système et système	Plan et procédures d'intégration/validation	
Gestion de configuration		Source Safe 5.0, OFFICE 97, EXCEL
Traitement de la documentation		OFFICE 97

Tableau 2 - Méthodes et Outils utilisés pour la réalisation du SD-SSALTO

3.2.2.1.Codage

Les normes de codage C++ qui sont appliquées au codage du logiciel serveur de données SSALTO, sont décrites dans le document Standard de codage.

Respect des normes de codage

Les standards de codage ont pour objet de garantir l'homogénéité et la maintenabilité de l'ensemble de code source développé ainsi que la sécurité des développements. L'objectif de la vérification du respect de ce standard est de s'assurer que cette homogénéité et cette sécurité ont été bien obtenue pour l'ensemble des développements. Ce type de contrôle a consisté en l'inspection manuelle d'échantillons de code pour vérification du respect des règles portant :

- sur le taux de commentaires,
- sur les instructions de code,
- sur le traitement des erreurs.

Ces contrôles ont été effectués par l'Ingénieur Qualité.

Un premier contrôle a été effectué en début de phase de codage sur les premières unités de programmation développées par chaque développeur. Le but de ce premier contrôle était d'évaluer au plutôt la connaissance et le respect des normes établies afin de :

- fournir immédiatement aux développeurs un retour sur les respect des normes,
- proposer s'il y a lieu des actions correctives pour réduire tout éventuel écart ayant été détecté.

Un contrôle plus large a ensuite été mené au cours de la phase, avant la livraison partielle des 30%.

Enfin, un dernier contrôle a été effectué en fin de phase de réalisation. Ce contrôle a porté sur des échantillons déterminés selon les règles décrites ci-dessus et n'ayant fait l'objet d'aucun contrôle.

Mesure de la complexité du code

Les mesures de complexité sur le code sont effectuées au moyen de l'outil Logiscope Code Checker, outil d'analyse statique de code source, disponible pour les langages C et C++.

Cet outil repose sur la définition de métriques qui sont mesurées lors de l'analyse et de valeurs de seuil associées afin d'obtenir une classification des composants analysés.

Les métriques définies dans le cadre du projet serveur de données de SSALTO portent sur la complexité des fonctions C et des méthodes C++ ainsi que sur le taux de commentaires. Pour chacune des métriques retenues, des seuils ont été établis, définissant l'intervalle des valeurs acceptables. Le tableau ci-dessous fait la synthèse des métriques retenues et des seuils associés :

Métrique	Description	Seuils	
		Min	Max
STMT	Nombre d'instructions exécutables comprises entre l'en-tête de la fonction et l'accolade terminale	1	50
VG	Nombre cyclomatique : nombre de chemins linéairement indépendant	1	15
AVGS	Taille moyenne des instructions (nombre moye, d'opérandes et d'opérateurs utilisés par chaque instruction exécutable de la fonction	1.00	7.00
COMF	Fréquence des commentaires	0.20	1
GOTO	Nombre d'instructions GOTO	0	0
LEVL	Nombre de niveau : nombre maximal d'imbrications des structures de contrôle	1	5

Tableau 3 - Métriques de SSALTO

L'analyse menée dans le cadre du projet a permis de confirmer la conformité globale du code avec ces règles. Ces métriques sont comparables à celles proposées pour l'OPS, qui sont résumées dans le tableau ci-dessous :

Métrique	Description	Seuils	
		Min	Max
NB_INS	Nombre d'instructions exécutables comprises entre l'en-tête de la fonction et l'accolade terminale	1	100
VG	Nombre cyclomatique : nombre de chemins linéairement indépendants	1	20
NB_NIV	Nombre de niveau : nombre maximal d'imbrications des structures de contrôle	1	6
T_COM	Taux de commentaires.	0.30	1.00

Tableau 4 - Métriques de l'OPS IASI

Dans le cas de divergences entre les métriques (ex : taux de commentaire), il n'est pas prévu de modifier le code réutilisé pour le mettre aux standards qualité de l'OPS IASI.

Le volume de code du logiciel a été lui aussi analysé et présenté dans le tableau suivant :

Eléments mesurés	Résultat sur la version 2.4
Nombre de répertoires	20
Nombre total de lignes	75662
Nombre d'instructions exécutables	41365

Tableau 5 - Volume de code de SSALTO

Le volume du code que nous envisageons de réutiliser est estimé à 20000 instructions exécutables.

3.2.2.2.Validation

La validation menée dans le cadre du serveur de données SSALTO a mis en évidence une cinquantaine de FA actuellement toutes corrigées. Depuis lors, le système est utilisé de façon opérationnelle au CNES.

Actuellement, aucune FA ouverte ne concerne les parties de composants que nous nous proposons de réutiliser (CMN et SD).

3.3.ANALYSE DES COMPOSANTS A REUTILISER

Le composant en charge de générer les produits IASI L1 de l'OPS est implémenté à l'aide d'un processus multi-threadé, basé sur les threads POSIX. La solution retenue est la réutilisation de l'architecture du processus SD du logiciel SD-SSALTO qui est basée sur l'implémentation suivante :

- un **thread principal** lance et supervise les différents traitements,
- un **thread de monitoring** traite les requêtes externes; dans le cas de l'OPS elles émanent du processus WOM,
- un ensemble (pool) de **threads de traitement** est dédiés aux tâches opérationnelles proprement dites. Dans le cas de l'OPS il s'agit d'enchaînement d'algorithmes. Ce nombre de thread est paramétrable au lancement du process.

Le schéma suivant présente l'architecture multi-threads du SD que nous proposons de mettre en place pour implémenter la parallélisation du traitement d'un granule.

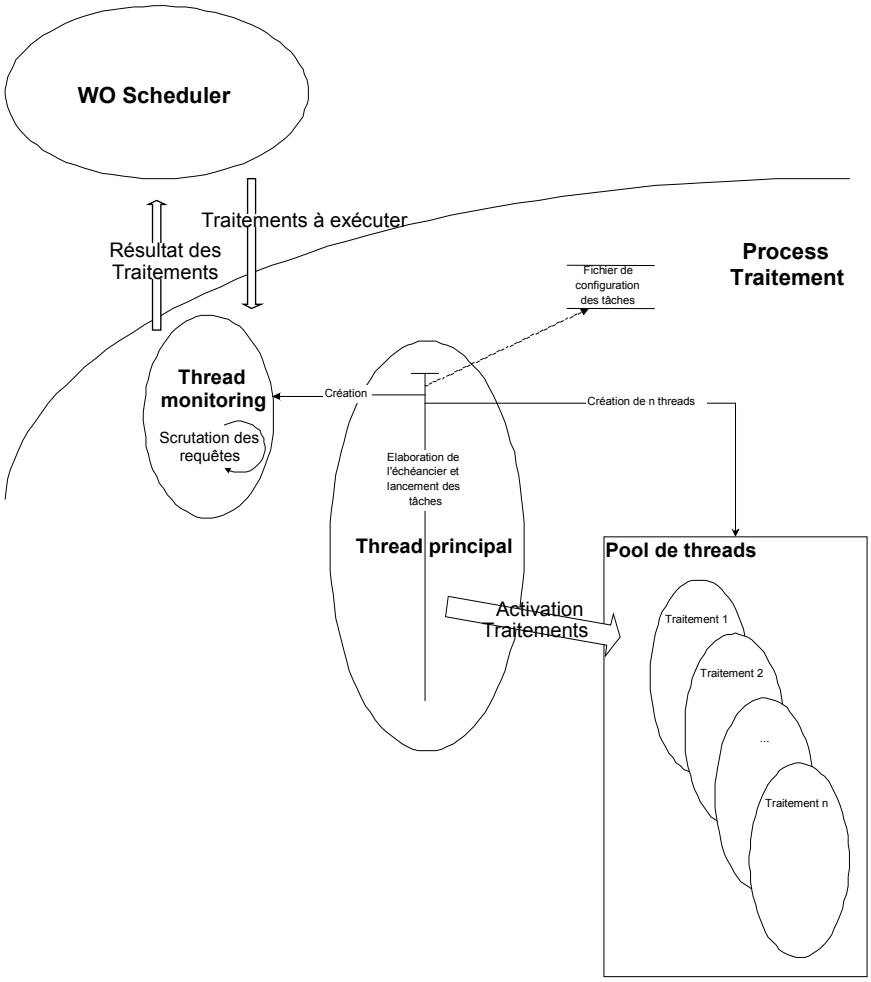

Figure 3 - Gestion Interne des Threads

3.3.1.Composant «CMN»

3.3.1.1.Rôle

Ce composant est constitué des modules communs, qui sont utilisé par le processus SD. Il comprend 11 sous-composants :

- ALM : modules de gestion des alarmes,
- BAL : modules de communication par boîte aux lettres,
- CFG : modules de gestion des fichiers de configuration,
- CRY : modules de cryptage des données,
- DAT : modules de données,
- EVT : modules de gestion des événements,
- GLB : modules de base,
- MES : modules de gestion des messages,
- OBS : modules de gestion des observateurs,
- SES : modules de gestion des sessions,
- TEC : modules de supervision des tâches.

3.3.1.2.Taux de réutilisation et axes de modification envisagés

Taux de réutilisation

Le tableau suivant liste les classes du composant CMN et indique pour chacune d'elle le taux de réutilisation envisagé.

Modules	Réutilisation (en %)
Sous-composant ALM	**0**
CMN_ALM_GestionnaireAlarmes	0
CMN_ALM_ReponseAcquitterAlarme	0
CMN_ALM_RequeteAcquitterAlarme	0
CMN_ALM_ReponseListerAlarmes	0
CMN_ALM_RequeteListerAlarmes	0
CMN_ALM_SignalerAlarme	0
CMN_ALM_SujetSurAlarme	0
Sous-composant BAL	**0**
CMN_BAL_BoiteALettre	0
Sous-composant CFG	**0**
CMN_CFG_Config	0

CMN_CFG_Configuration	0
Sous-composant CRY	**0**
CMN_CRY_Cryptage	0
CMN_CRY_Des	0
Sous-composant DAT	**20**
CMN_DAT_AccesSite	0
CMN_DAT_Alarme	0
CMN_DAT_Ecart	0
CMN_DAT_EspaceSD	0
CMN_DAT_Evenement	90
CMN_DAT_ListeObjets	100
CMN_DAT_ModeleAssociation	0
CMN_DAT_ModeleTache	60
CMN_DAT_Objet	100
CMN_DAT_Periodicite	0
CMN_DAT_Rattachement	0
CMN_DAT_SystemeDeFichier	0
CMN_DAT_Tache	0
CMN_DAT_Transfert	0
CMN_DAT_TransfertExport	0
CMN_DAT_TransfertImport	0
CMN_DAT_TypeAlarme	0
CMN_DAT_TypeDeFichier	0
CMN_DAT_Utilisateur	0
Sous-composant EVT	**0**
CMN_EVT_FiltreEvenement	0
CMN_EVT_GestionnaireEvenements	0
CMN_EVT_ReponseListerEvenements	0
CMN_EVT_RequeteListerEvenements	0
CMN_EVT_SujetSurEvenement	0
CMN_EVT_SignalerEvenement	0
Sous-composant GLB	**0**
CMN_GLB_Chaine	0
CMN_GLB_Date	0
CMN_GLB_Fichier	0
CMN_GLB_Repertoire	0
CMN_GLB_Securite	0
CMN_GLB_Trace	0
CMN_GLB_Version	0

Sous-composant MES	**0**
CMN_MES_LancerAdm	0
CMN_MES_Message	0
CMN_MES_ReponseConfigurer	0
CMN_MES_RequeteConfigurer	0
CMN_MES_RequetePurgerEspaceDisque	0
CMN_MES_RequeteVerifierEspaceDisque	0
Sous-composant OBS	**100**
CMN_OBS_Observateur	100
CMN_OBS_Sujet	100
Sous-composant SES	**0**
CMN_SES_ReponseConnecter	0
CMN_SES_ReponseDeconnecter	0
CMN_SES_RequeteConnecter	0
CMN_SES_RequeteDeconnecter	0
CMN_SES_RequeteMajEcartTAITUC	0
CMN_SES_RequeteMajRattachementBMTAI	0
CMN_SES_Session	0
Sous-composant TEC	**0**
CMN_TEC_GestionnaireTachesEnCours	0
CMN_TEC_ReponseListerTaches	0
CMN_TEC_RequeteArreterTache	0
CMN_TEC_RequeteDemarrerTache	0
CMN_TEC_RequeteListerTaches	0
CMN_TEC_SignalerEtatTache	0
CMN_TEC_SujetSurTacheEnCours	0
TOTAL Composant CMN	**10**

Tableau 6 - Taux de réutilisation du composant CMN de SSALTO

CMN_DAT_EVENEMENT : uniquement mise à jour du code pour intégrer le nouveau format des messages JdB

CMN_DAT_ModeleTache : classe sans « intelligence » servant uniquement a stocker une structure de données

Axes de modification

Le composant CMN sera enrichi de nouveaux modules :

- pour gérer la communication par message avec les interfaces externes,
- pour implémenter les tâches exécutées par les threads propres à l'OPS IASI. Ces tâches seront dérivées des tâches génériques que gère le SD.

Les évolutions des modules existants respecteront les standards de codage en vigueur sur SSALTO.

3.3.2.Composant «SD»

3.3.2.1.Rôle

Le composant SD est un processus multi-threadé (POSIX) qui gère un échéancier (matérialisé par un fichier) regroupant toutes les tâches à exécuter par l'application. Un pool de threads configurable est initialisé et géré par le thread principal ; un dernier thread s'occupe de la prise en compte des demandes d'exécution de tâches émanant d'une interface externe.

Le processus SD est lancé à la fin de la phase de boot du système. Il ne s'arrête jamais. Le thread principal du processus s'appuie sur un pool de threads non spécialisés pour exécuter différentes tâches (importation/exportation de fichiers, purge, surveillance de l'espace disque, etc …). Les threads sont lancés au début du processus. Ils ne s'arrêtent jamais, car dès qu'un thread a fini d'exécuter une tâche, il redevient disponible pour exécuter une autre tâche.

Les tâches informatiques exécutées par les threads de traitement sont décrites dans un fichier de configuration.

Le composant SD formera le cœur du process de Traitement de l'OPS IASI. Actuellement, il comprend 8 sous-composants :

- ACT : modules de gestion des actions,
- CMD : modules de gestion de commandes et de comptes-rendus,
- DAT : modules des données propres au SD,
- EVT : modules de gestion des événements du SD,
- FRW : modules de base du serveur SD,
- SRV : modules des serveurs associés au SD,
- TAC : modules de gestion des tâches.

Taux de réutilisation et axes de modification envisagés

Taux de réutilisation :

Modules	Réutilisation (en %)
Sous-composant ACT	**9**
SD_ACT_Action	80
SD_ACT_ActionDeControle	0
SD_ACT_ActionDeTest	100
SD_ACT_ActionDeTransfert	0
SD_ACT_CompteRenduEtatServeur	0
SD_ACT_ControleBlocFin	0
SD_ACT_ControleNbPoussee	0
SD_ACT_ControleSequenceEntete	0
SD_ACT_ControleVolume	0
SD_ACT_Export	0
SD_ACT_ExportAsynchrone	0
SD_ACT_ExportSynchrone	0
SD_ACT_HistoCentreDeMasseEnvisat	0
SD_ACT_HistoCentreDeMasseJason	0
SD_ACT_HistoCentreDeMasseSpot	0
SD_ACT_HistoEcartTaiTuc	0
SD_ACT_HistoManoeuvreEnvisatMCSF	0
SD_ACT_HistoManoeuvreEnvisatMOHF	0
SD_ACT_HistoManoeuvreJasonOEF	0
SD_ACT_HistoManoeuvreJasonOPF	0
SD_ACT_HistoManoeuvreSpot	0
SD_ACT_HistoRattachementBmTai	0
SD_ACT_Historisation	0
SD_ACT_Import	0
SD_ACT_ImportAsynchrone	0
SD_ACT_ImportSynchrone	0
SD_ACT_ProduireManoeuvreIGN	0
SD_ACT_PurgeSurSA	0
SD_ACT_PurgeSurSD	0
SD_ACT_RechercheDeFichiers	0
SD_ACT_RenommageSATD	0

Modules	Réutilisation (en %)
SD_ACT_RenommageSATDAussaguel	0
SD_ACT_RenommageSATDKiruna	0
SD_ACT_SurveillanceEspace	0
SD_ACT_Traduction	0
SD_ACT_Traitement	100
SD_ACT_TraitementBmTai	0
SD_ACT_TransfertEspaceExterneVersInterne	0
SD_ACT_TransfertEspaceInterneVersExterne	0
SD_ACT_VerificateurSymptome	100
Sous-composant CMD	**0**
SD_CMD_CommandeArretSa	0
SD_CMD_CommandeArretTache	0
SD_CMD_CommandeEtatServeur	0
SD_CMD_CommandeEtatTache	0
SD_CMD_CommandeExportDistant	0
SD_CMD_CommandeExportLocal	0
SD_CMD_CommandeImportDistant	0
SD_CMD_CommandeImportLocal	0
SD_CMD_CommandePurge	0
SD_CMD_CompteRendu	0
SD_CMD_CompteRenduArretSa	0
SD_CMD_CompteRenduArretTache	0
SD_CMD_CompteRenduEchecSysteme	0
SD_CMD_CompteRenduEtatServeur	0
SD_CMD_CompteRenduEtatTache	0
SD_CMD_CompteRenduExportLocal	0
SD_CMD_CompteRenduFinalExportDistant	0
SD_CMD_CompteRenduFinalImportDistant	0
SD_CMD_CompteRenduImportLocal	0
SD_CMD_CompteRenduPartielExportDistant	0
SD_CMD_CompteRenduPartielImportDistant	0
SD_CMD_CompteRenduPurge	0
SD_CMD_ReconfigurationDesSites	0
Sous-composant DAT	**0**
SD_DAT_Echeance	0
Sous-composant EVT	**95**
SD_EVT_ObservateurTerminaisonTache	90

Modules	Réutilisation (en %)
SD_EVT_SujetTerminaisonTache	100
Sous-composant FRW	**70**
SD_FRW_ApplicationServeurDonnees	70
SD_FRW_ContexteThread	95
SD_FRW_Echeancier	30
SD_FRW_FabriqueAction	90
SD_FRW_FabriqueDeTache	90
SD_FRW_GestionnaireThread	90
SD_FRW_VerrouApplication	100
Sous-composant SRV	**40**
SD_SRV_ServeurAlarmes	0
SD_SRV_ServeurEvenements	80
Sous-composant TAC	**40**
SD_TAC_ContexteDeTache	30
SD_TAC_GestionnaireDeSession	60
SD_TAC_Tache	90
SD_TAC_TacheTraitement	90
TOTAL Composant CMN	**55**

Tableau 7 - Taux de réutilisation du composant SD de SSALTO

SD_ACT_ACTION : modification porte pour l'essentiel sur la suppression de méthodes externes inutiles dans le cadre de l'OPS. Le code restant est réutilisé à 95%.

Axes de modification :

Le cœur du composant SD ne bougera pas sauf l'échéancier qui doit légèrement évoluer :

- suppression de la notion d'échéance datée pour les taches, l'échéancier dans le cas de l'OPS devient une liste de type FIFO,
- rajout de la notion de rendez-vous entre tâches.

Les actions applicatives spécifiques à SSALTO (import/export, ..) gérées par le composant SD sont supprimées et remplacées par des actions applicatives de l'OPS (enchaînement d'algorithmes).

Les outils et shell seront réutilisés après adaptation.

Afin que le SD s'insère dans l'architecture du CGS des adaptations sont par ailleurs nécessaires pour implémenter les besoins suivants :

- le démarrage du processus par le Main Process au lieu du démarrage au boot de la machine,
- la communication avec les interfaces externes pour échanger de nouveau type d'information (demande de traitement, HKTM, …).

4.REUTILISATION DU SIF

4.1.PRESENTATION DU LOGICIEL

4.1.1.Fonctionnalités

Le SIF est un logiciel du F-PAC chargé d'interfacer le Centre de Mission ENVISAT et la composante GDD du système SSALTO du CNES. Son rôle est de simuler du point de vue du centre de mission ENVISAT les sous-systèmes ARF et PF-HS d'ENVISAT absents dans le F-PAC du CNES. Il doit à ce titre assurer les fonctions suivantes :

- traiter des instructions en provenance du CMC (Control & Monitoring Center) : traduire ces requêtes en ordres pour le GDD chargé de les exécuter, ou exécuter lui-même ces requêtes s'il en a la capacité,
- maintenir à jour l'inventaire des données et produits disponibles au GDD,
- mettre à disposition du GDD les produits transmis par le DF (Dissemination Facility),
- transmettre au DF les produits reçus du GDD pour diffusion aux utilisateurs,
- mettre à disposition de l'USF les petits produits transmis par le GDD, pendant une période limitée,
- envoyer périodiquement son état fonctionnel au CMC,
- mettre périodiquement à disposition du GDD un journal de bord spécifique, distinct du journal de bord ENVISAT, récapitulant les différents événements survenus au cours des traitements du SIF.

4.1.2.Conditions de Réalisation

4.1.2.1.Développement du Produit

Le SIF a été réalisé par THALES IS en 2000 pour le compte du CNES.

La structure du produit est décrite dans le document de Conception Préliminaire du SIF.

Le SIF est considéré comme un sous-système de la composante F-DAC du F-PAC.

L'organigramme du produit SIF présenté ci-dessous montre l'arborescence des unités de réalisation constituant le sous-système SIF.

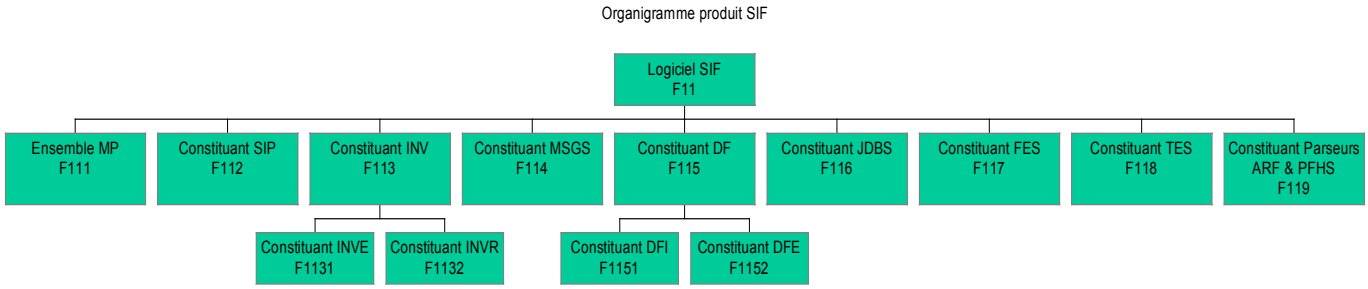

Figure 4 : Organigramme produit SIF

Les éléments marquant du développement, sont les suivants :

- cycle de développement court (7,5 mois calendaires),
- spécifications finalisées dès l'AO, donc stables (une phase de spécification classique n'a pas été nécessaire),
- couplage faible d'un point de vue informatique (transfert de fichiers par ftp) du SIF avec les Eléments Génériques du F-DAC, ce qui signifie une robustesse implicite de l'architecture du SIF vis à vis de modifications au niveau des interfaces,
- le SIF ne présente pas d'IHM.

Compte tenu de ces caractéristiques, la démarche de développement adoptée pour la réalisation de ce produit logiciel a consisté en un cycle de développement en V.

4.1.2.2.Condition de garantie et de maintenance

Ce système est actuellement maintenu sur le site de THALES IS à Toulouse.

La dernière version du SIF est la V2.5.

Depuis la recette site de la V1.0 du SIF (septembre 2000), 15 anomalies ont été levées et 14 sont closes. La dernière anomalie, levée mi avril 2001, est en cours de correction, elle concerne une fonction spécifique du SIF sans rapport avec la partie du logiciel réutilisé dans le cadre de IASI.

4.1.3.Documentation associée

Le projet SIF a été réalisé sur la base des exigences générales décrites dans le document « Exigences de réalisation des composants du SSALTO hors classe 1 » (SMM-ST-M-GO-20065-CN Ed. 1 Rev 1). La documentation projet est néanmoins claire et suffisante, elle est résumée dans le tableau ci-dessous :

Référence	Intitulé
FPA-SL-F11-EA-20651-SYS	Spécification Logicielle du SIF
FPA-CP-F11-EA-20652-SYS	Conception Préliminaire du SIF.
FPA-CD-F11-EA-20664-SYS	Conception Détaillée
FPA-PG-F11-EA-20653-SYS	Plan d'application du SIF
FPA-AQ-F11-EA-20655-SYS	Bilan qualité
FPA-NQ-F11-EA-20659-SYS	Standard de Codage
FPA-PG-F11-EA-20656-SYS	Plan de gestion de configuration
FPA-PG-F11-EA-20657-SYS	Plan d'assurance sécurité
FPA-PE-F11-EA-20660-SYS	Plan des essais d'intégration et de validation
FPA-MU-F11-EA-20661-SYS	Manuel d'utilisation
FPA-VR-F11-EA-20662-SYS	Cahier de recette
FPA-MI-F11-EA-20666-SYS	Manuel d'installation
FPA-MM-F11-EA-20669-SYS	Manuel d'aptitude à la maintenance

Tableau 8 - Documentation du Projet SIF

4.1.4.Environnement matériel et logiciel

4.1.4.1.Environnement matériel

Le SIF a été développé sur station IBM RS6000 43P sous AIX 4.2 en utilisant le langage C++ (codage, compilation, déboggage).

La conception a été réalisée en utilisant une approche OBJET formalisée grâce à l'outil Rational Rose / UML (version 98).

La gestion de configuration est effectuée via l'outil CVS.

4.1.4.2.Architecture logicielle

La figure suivante présente l'architecture logicielle du SIF :

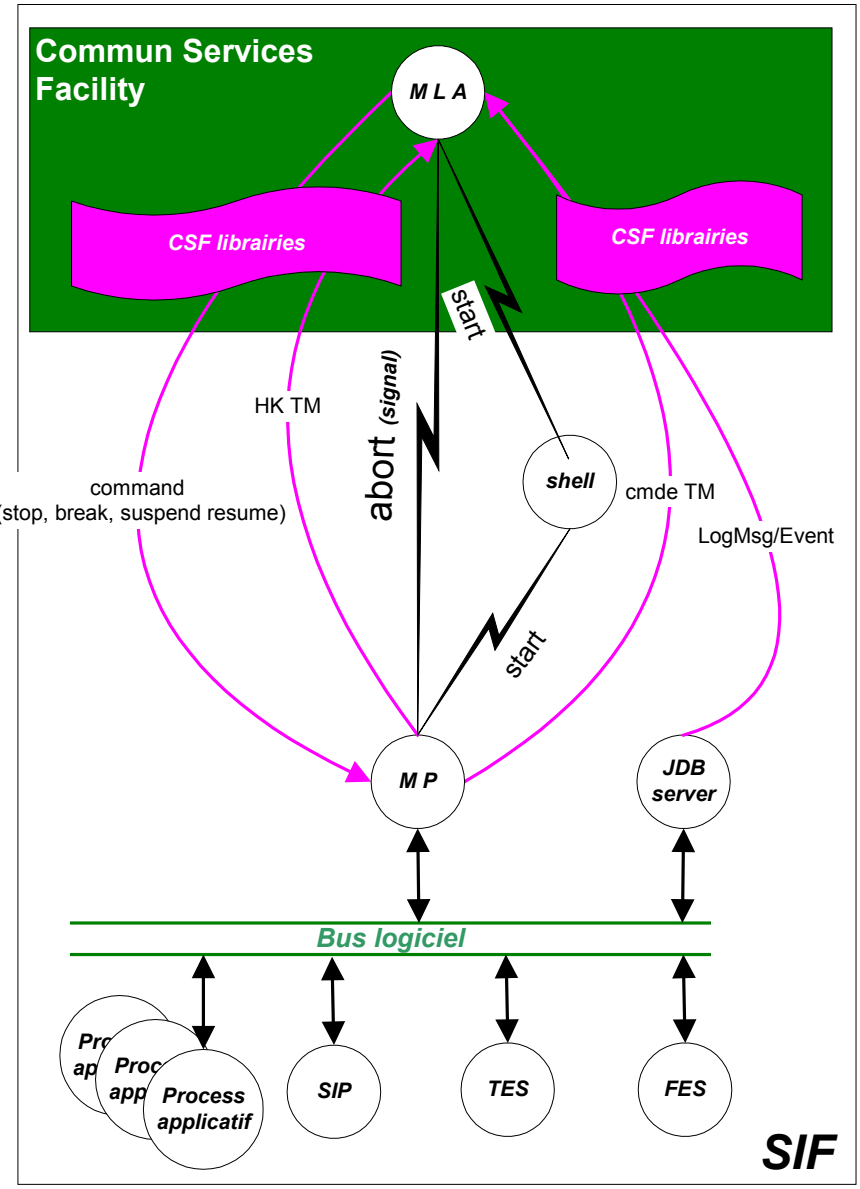

Figure 5 - Architecture Logicielle du SIF

L'architecture du SIF est construite autour d'un *Bus Logiciel* qui encapsule et banalise les communications inter-process. Ce bus utilise le mécanisme traditionnel de communication par socket disponible sur tous les systèmes UNIX ainsi que sous Windows. Ce bus est basé sur la distribution de messages à des adresses logiques auxquelles se sont abonnés des clients. Lorsqu'un message est émis sur le bus, tous les process abonnés à ce type de message le reçoivent. Le SIF s'interface avec le système de Monitoring & Control (MCS) du PDS-ENVISAT à l'aide du process MLA à travers les librairies CSF (Common Services Facility). Ces librairies regroupent les méthodes qui implémentent les échanges de données entre le SIF et le MLA.

Le SIF est composé des process suivants :

- le *Main Process* (MP) qui a en charge le démarrage, la surveillance et l'arrêt de tous les process. De plus, ce process est le point unique et obligatoire (imposé par l'architecture) d'interfaçage entre le SIF et le MCS,
- le SIP (Sub Instruction Performer) est le process en charge de traiter et exécuter les ordres de production reçus du MCS. Dans le cadre du SIF, ces ordres sont de type transfert de données entre sous-systèmes,
- le TES (Time Event Server) est le process en charge de gérer les événements "Timer" que les processus applicatifs ont programmés. Dès que la date est atteinte, le processus applicatif ayant programmé cette date est prévenu,
- le FES (File Event Server) est un process en charge de scruter les répertoire que les processus applicatifs ont programmé. Dès qu'un nouveau fichier est présent sous un des répertoires scruter le processus applicatifs ayant programmé la scrutation est prévenu.,
- le JDB Server est en charge de collecter et transmettre les messages JdB. Ce process permet d'encapsuler l'interface JdB, ce qui facilite le portage du socle SIF,
- les process applicatifs propres aux sous-systèmes PDS à simuler (ARF et PF-HS).

4.2.INTERET DE LA REUTILISATION

4.2.1.Adéquation aux besoins techniques

Nous proposons de réutiliser le socle commun du SIF pour implémenter la fonctionnalité de Monitoring & Control de l'OPS IASI. Cette solution est sur-dimmensionnée étant donné que les fonctionnalités offertes par le SIF sont plus étendues que le besoin de l'OPS IASI. Mais elle présente les avantages suivants :

- sécuriser le développement logiciel de l'OPS par la réutilisation d'un logiciel robuste et fiable : le SIF fonctionne 24 heures sur 24 et à fait l'objet de peu d'anomalies logicielles. Il apporte d'ailleurs toute satisfaction au CNES,
- diminuer le coût de développement de l'OPS IASI. La réutilisation et l'adaptation du SIF permettent une économie substantielle sur les phases de conception, développement et validation. Cette solution permet de concentrer les efforts lors de ces phases sur les traitements algorithmiques et plus particulièrement les problèmes liés à l'optimisation des performances,
- d'offrir des services de base utiles pour l'OPS IASI.

De part sa disponibilité et sa simplicité d'utilisation, nous proposons en particulier de conserver le bus de communication inter-process présenté au § 4.1.4.2 dans l'architecture de l'OPS IASI.

L'architecture logicielle de l'OPS IASI dérivant du SIF que nous proposons est présentée sur la figure suivante :

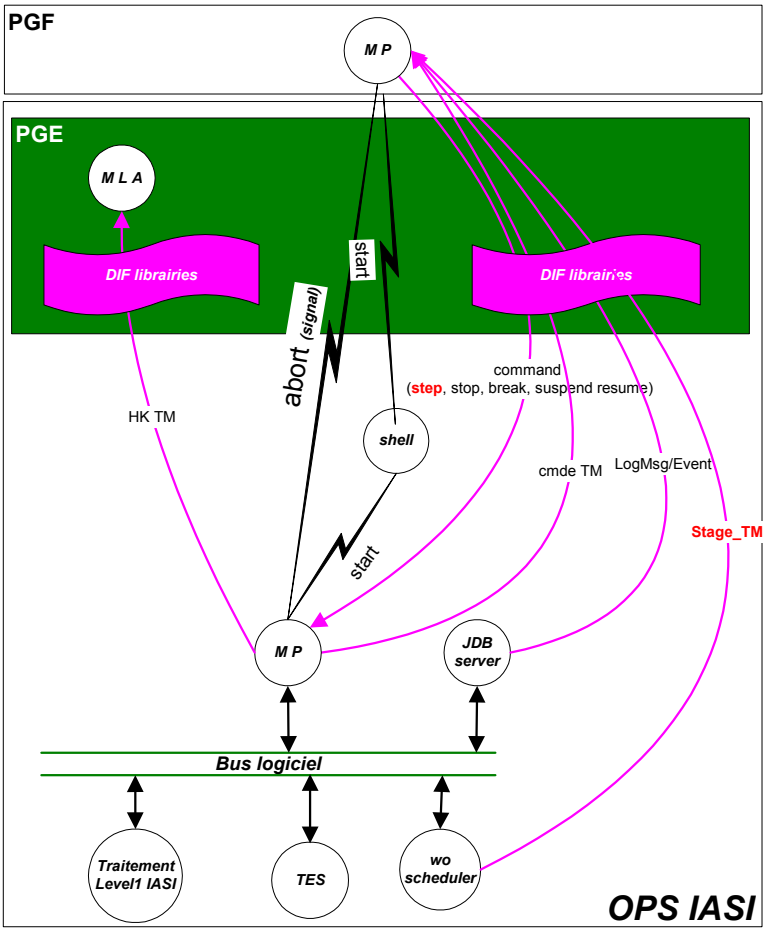

Figure 6 - Architecture Logicielle de l'OPS IASI

La stratégie de réutilisation est la suivante :

- les processus du SIF shell de démarrage, MP, TES et JdB server ainsi que le Bus Logiciel sont conservés,
- le process WO Manager (traitement des work orders) remplace le processus SIP (traitement des instructions) du SIF,
- le process Traitement IASI Level1 qui est spécifique est rajouté dans l'architecture du SIF,
- le process FES est supprimé étant donné que l'OPS IASI ne nécessite pas de déclencher des tâches d'arrivée de fichier.

4.2.2.Adéquation aux besoins qualités

Le logiciel SIF a été développé en respectant des règles strictes de qualité répertories dans son Plan d'application. Ces besoins sont très proches des contraintes imposées sur le système IASI OPS dans le cadre des marchés de classe 1. En particulier les méthodes et outils utilisés dans les différentes phases de développement sont résumés ci dessous :

PHASE	METHODE	OUTIL
Conception préliminaire	Méthode orientée Objet Notation UML	Rational Rose
Conception détaillée*	Raffinement des classes Entête de classe et de méthode + Pseudo code	Outil d'extraction de la CD à partir des sources (déjà utilisé dans le cadre du SD SSALTO).
Codage	MPM C++ Normes de codage C++	C++ LINT INSURE++ Logiscope statique (C++ Code checker)
Tests Unitaires	Stratégie de test unitaire décrite dans le Dossier de définition des composants.	
Intégration/validation sous-système et système	Plan et procédures d'intégration/validation	
Gestion de configuration		CVS, OFFICE 97, EXCEL
Traitement de la documentation		OFFICE 97

Tableau 9 - Méthodes et Outils utilisés pour la réalisation du SIF

4.2.2.1.Codage

Les normes de codage C++ qui sont appliquées au codage du SIF, sont décrites dans le document Standard de codage du SIF.

Les mesures de complexité sur le code sont effectuées au moyen de l'outil Logiscope Code Checker. Cet outil permet de réaliser des analyses statiques de codes source écrits en langage C++.

Cet outil repose sur la définition de métriques qui sont mesurées lors de l'analyse et de valeurs de seuil associées afin d'obtenir une classification des composants analysés.

Les métriques définies dans le cadre du projet SIF portent sur la complexité des méthodes C++.

Elles sont présentées dans le tableau ci-dessous :

Métrique	Description	Seuils	
		Min	Max
STMT	Nombre d'instructions exécutables comprises entre l'en-tête de la fonction et l'accolade terminale	1	60
VG	Nombre cyclomatique : nombre de chemins linéairement indépendants	1	17
AVGS	Taille moyenne des instructions (nombre moyen, d'opérandes et d'opérateurs utilisés par chaque instruction exécutable de la fonction	1.00	7.00
COMF	Fréquence des commentaires	0.20	1
GOTO	Nombre d'instructions GOTO	0	0
LEVL	Nombre de niveau : nombre maximal d'imbrications des structures de contrôle	1	5

Tableau 10 - Métriques du SIF

L'analyse menée dans le cadre du projet a permis de confirmer la conformité globale du code avec ces règles.

Ces métriques sont comparables à celles proposées pour l'OPS, qui sont résumées dans le tableau ci-dessous :

Métrique	Description	Seuils	
		Min	Max
NB_INS	Nombre d'instructions exécutables comprises entre l'en-tête de la fonction et l'accolade terminale	1	100
VG	Nombre cyclomatique : nombre de chemins linéairement indépendants	1	20
NB_NIV	Nombre de niveau : nombre maximal d'imbrications des structures de contrôle	1	6
T_COM	Taux de commentaires.	0.30	1.00

Tableau 11 - Métriques de l'OPS IASI

Dans le cas de divergences entre les métriques (ex :taux de commentaire), il n'est pas prévu de modifier le code réutilisé pour le mettre aux standards qualité de l'OPS IASI.

Le volume de code du logiciel a été lui aussi analysé et est synthétisé dans le tableau suivant :

Eléments mesurés	Résultats sur la version initiale	Résultat sur la version 1.3
Nombre de répertoires (processus)	13	14
Nombre de classes C++	60	73
Nombre de méthodes	562	1058
Nombre total de lignes	38 483	66 516
Nombre d'instructions exécutables	7 837	10 957

Tableau 12 - Volume de code du SIF

Le volume du code que nous envisageons de réutiliser est estimé à 7700 instructions exécutables.

4.2.2.2.Validation

La validation menée dans le cadre du SIF a mis en évidence un faible niveau d'anomalie qui en outre sont en majorité issues du processus SIP non réutilisé dans l'OPS IASI.

De plus, aucune anomalie n'a été détectée concernant les processus TES, JDBS et MSGS. Ceci illustre le niveau de fiabilité du système, ce qui a été confirmé par l'utilisation opérationnelle au CNES.

4.3.ANALYSE DU LOGICIEL A REUTILISER

4.3.1.Composant MSGS « MeSsaGe Server »

4.3.1.1.Rôle

Le serveur de messages est le socle de l'architecture du SIF. C'est un processus de service de l'architecture SIF. Il permet de banaliser la communication inter-processus et de créer un bus logiciel sur lequel viennent se connecter les autres processus du SIF. Le mécanisme de base utilisé dans ce cadre est le socket, l'architecture logique choisie est illustrée dans le schéma ci-dessous.

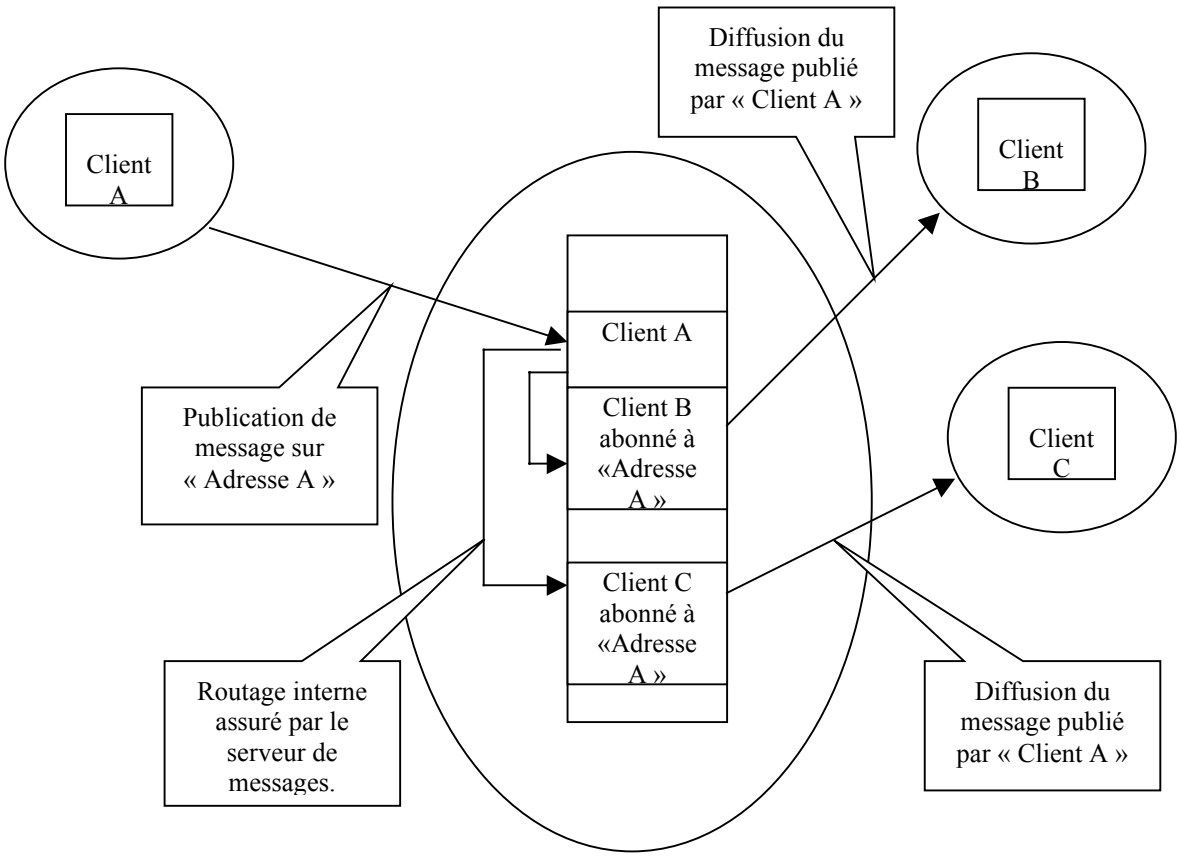

Figure 7 - Routage des adresses logiques

Les différents clients connectés au « serveur de messages » s'échangent des messages. On peut distinguer deux types de messages selon la sémantique qu'ils transportent :

- les « messages de commandes »qui sont échangés entre les « processus de services »,
- les « messages applicatifs » dont la sémantique est libre et fixée par convention entre un producteur et un (ou N) consommateurs.

Le « serveur de messages » (MSGS pour MeSsaGe Server) est un processus permanent qui est lancé par le Main Process (contrainte CSF) dés le démarrage du SIF.

4.3.1.2.Taux de réutilisation et axes de modification envisagés

Dans le cadre de l'OPS IASI le Serveur de message est repris intégralement car il est l'épine dorsale du système et permettra de standardiser la communication entre les différents processus de l'OPS.

Taux de réutilisation :

Modules	Réutilisation (en %)
Sous-composant MSG	**100**
MSG__Abonnement	100
MSG__Connexion	100
MSG__Socket	100
Sous-composant MSGS	**100**
MSGS__Adresse	100
MSGS__GestionnaireActivite	100
MSGS__ServeurMessage	100
TOTAL Composant MSGS	**100**

Tableau 13 - Taux de réutilisation du composant MSGS du SIF

4.3.2.Composant MP « Main process »

4.3.2.1.Rôle

Le « processus principal » MP (Main Process) est chargé de faire l'interface entre le MPA (Main Process Agent) qui assure la collecte du flot de Monitoring & Control et le remonte au CMC

Le MP a la responsabilité :

1. de nettoyer au démarrage les fichiers « instruction CMC », « Rapports de sous-instructions »,
2. de lancer tous les processus nécessaires pour assurer la fonctionnalité du sous-système,
3. de détecter et d'analyser les fichiers « instructions CMC »,
4. de renvoyer les « acquittements d'instruction CMC »,
5. de renvoyer les « acquittements de sous-instruction CMC »,
6. de renvoyer les « rapports de sous-instruction »,
7. de collecter et de renvoyer les « status de ressources »,
8. d'effectuer la purge des petits produits USF.

Le MP est démarré soit par le MPA, soit en local sur la machine, par un shell script spécifique.

Les messages applicatifs reçus par le MP sont :

- message de type SUB_INSTRUCTION_STATUS qui permet au processus « exécuteur de sous-instruction » de transmettre l'évolution de la sous-instruction. Ce message contient l'identificateur de la sous-instruction et le status associé,
- message de type SUB_INSTRUCTION_REPORT qui permet au processus « exécuteur de sous-instruction » de soumettre au MP le rapport de sous-instruction à renvoyer. Ce message contient l'identificateur de la sous-instruction, et le nom du fichier qui contient le rapport,
- message de type RESSOURCE_STATUS qui contient le nom du « status de ressources » ainsi que sa valeur courante.

Les messages applicatifs envoyés par le MP sont :

- Un message de commande de type EVT_FIN_EXECUTION pour signaler à tous les autres processus une fin d'exécution demandée par le CMC ou en local par l'exploitant du SIF.

4.3.2.2.Taux de réutilisation et axes de modification envisagés

L'architecture de Monitoring & Control de l'OPS IASI est proche à celle du SIF; elle est basée sur 2 interfaces :

- une API de surveillance avec un processus du sous-système de Monitoring & Control (le CMC pour SIF, le MCS pour l'OPS) appelée MLA. La principale différence entre les 2 systèmes est que cette API est utilisée uniquement dans le cas de l'OPS pour l'envoi des Statuts et de messages JdB. Les services de cette API ont été renommés (leur signature étant pour la plus part identique).
- une API de commande différente de celle du SIF. En effet le SIF est commandé directement par le CMC alors que l'OPS est commandé par le PGF. Aucun service d'accès à cette API n'est fournie (c'était le cas pour le SIF); les services d'accès doivent être développés dans le cadre de l'OPS.

Taux de réutilisation

Modules	Réutilisation (en %)
MP__GestionnaireActivite	50
MP__GestionnaireStatus	95
MP__MainProcess	90
TOTAL Composant MP	**75**

Tableau 14 - Taux de réutilisation du composant MP du SIF

Axes de modification

Les évolutions pour adapter le shell et le MP du SIF à IASI sont minimes, elles concernent les points suivants :

- la prise en compte des arguments de lancement spécifiques à IASI : root directory et mode,
- la gestion de la nouvelle commande *STEP*,
- la mise à jour du fichier de configuration du MP (liste des processus du sous-système),
- la gestion des statuts HKTM spécifique au CGS.

Le Main Process OPS IASI est lancé via un shell par le MLA PGF. Ce processus a en charge le lancement de tous les processus du sous-système. Dans le cadre de l'OPS IASI, tous les processus sont permanents.

Le MP implémente les fonctions suivantes :

- le lancement des processus de l'OPS IASI,
- la réception et la diffusion des commandes du PGF. Les commandes sont analysées et diffusées sur le bus logiciel. Le traitement de chaque commande donne lieu à un compte rendu (Cmde_TM) renvoyé au PGF,
- l'envoi périodique ou sur événement de la HK TM (la périodicité de chaque type de TM est configurable),
- l'arrêt de tous les processus de l'OPS IASI sur la réception de la commande *STOP*. Cette commande est diffusée à tous les processus en exécution.

4.3.3.Composant TES « Time Event Server »

4.3.3.1.Rôle

Le Serveur d'Evénement Timer (le Timer Event Server TES) est un « processus de service » de l'architecture SIF. Son rôle est d'enregistrer, de générer, et de distribuer, en s'appuyant sur le serveur de messages, des événements représentant le déclenchement d'un timer logique. Les événements sont préalablement programmés par les processus applicatifs qui ont besoin d'être cadencés par le temps (par exemple voir chaque minute si un fichier instruction est arrivé).

4.3.3.2.Taux de réutilisation et axes de modification envisagés

Dans le cadre de l'OPS IASI, ce processus est réutilisé à 100% car il est utilisé par le MP pour envoyer de manière périodique la HKTM au PGF.

Taux de réutilisation :

Modules	Réutilisation (en %)
TES__GestionnaireActivite	100
TES__GestionnaireEvenement	100
TES__ServeurTemps	100
TOTAL Composant TES	**100**

Tableau 15 - Taux de réutilisation du composant TES du SIF

4.3.4.Composant JDBS « JdB Server »

4.3.4.1.Rôle

Le « Serveur JDB » (JDBS pour JDB Serveur) permet de gérer le JDB du FPAC, Celui-ci à pour vocation de recevoir tous les messages envoyés par les différents processus sur l'adresse logique « MessageJDB ». Le Serveur collecte tous les messages qui lui parviennent et gère un fichier temporaire qui contient les messages pour une durée de 5 minutes (configurable). Quand les 5 minutes sont écoulées, le fichier local est déplacé dans le répertoire « Message JDB GDD » et un nouveau fichier temporaire est créé pour les 5 minutes suivantes.

Le processus JDBS est un processus permanent qui est lancé par le Main Process (contrainte CSF) dés le démarrage du SIF.

Le journal de bord du SIF doit émettre ces messages vers deux destinations différentes :

- le CMC en utilisant le service commun **CSlogs** pour le journal de bord ENVISAT,
- des fichiers JDB FPAC à destination du GDD.

Les messages JDB sont stockés dans des fichiers configurables sans recompiler l'application.

4.3.4.2.Taux de réutilisation et axes de modification envisagés

Taux de réutilisation

Modules	Réutilisation (en %)
JDBS__GestionnaireActivite	90
JDBS__JDBServer	95
TOTAL Composant JDBS	**97**

Tableau 16 - Taux de réutilisation du composant JDBS du SIF

Axes de modification :

Ce process a en charge de collecter les messages JdB envoyé par l'ensemble des processus et de les renvoyer via une librairie du PGE.

Ce processus doit être adapté afin :

- de gérer 2 types de messages les log et les traces,
- d'utiliser les fonctions d'envoi de messages spécifique au CGS : LogEvent et LogTrace fournie par le PGE,
- d'être conforme au format des messages du CGS.

4.3.5.Composant CMN « Couche Support »

4.3.5.1.Rôle

Ce composant regroupe l'ensemble des services communs offerts aux processus du SIF.

Chaque module fournit les services liés à une encapsulation ou à une abstraction. Ceci est résumé dans le tableau ci-dessous :

Composant	Classe
CMN	
	CFG__Configuration
	CMN__Chaine
	CMN__Config

	CMN__Configuration
	CMN__Date
	CMN__DatePDS
	CMN__Fichier
	CMN__Rapport
	CMN__Securite
	CMN__Trace
	CMN__Version
	DFE__Instruction
	ERR__Exception
	FES__Evenement
	JDB__Callback
	JDB__Client
	JDB__Message
	MP__RapportSI
	MP__StatusRessource
	MP__StatusSI
	MP__UnProcess
	TES__Evenement

Tableau 17 - Classes du composant CMN du SIF

4.3.5.2.Taux de réutilisation et axes de modification envisagés

Taux de réutilisation

Modules	Réutilisation (en %)
Sous-composant CMN	**80**
CFG__Configuration	100
CMN__Chaine	100
CMN__Config	100
CMN__Configuration	100
CMN__Date	100
CMN__DatePDS	0
CMN__Fichier	100
CMN__Rapport	0
CMN__Securite	100
CMN__Trace	100

Modules	Réutilisation (en %)
CMN__Version	100
DFE__Instruction	0
ERR__Exception	40
FES__Evenement	0
JDB__Callback	70
JDB__Client	100
JDB__Message	100
MP__RapportSI	0
MP__StatusRessource	0
MP__StatusSI	0
MP__UnProcess	100
TES__Evenement	100
Sous-composant ENV	**100**
FMES_FICHIER_MESSAGES	100
FPARA_FICHIER_PARAMETRES	100
JDB_JOURNAL_DE_BORD	100
TOTAL Composant CMN	**85**

Tableau 18 - Taux de réutilisation du composant CMN du SIF

ERR_Exception : ce composant regroupe la liste de toutes les exceptions traitées dans l'OPS.

Axes de modification

Dans le cadre de l'OPS IASI, ce composant est réutilisé et adapté pour tenir compte des API des librairies du PGE et du format des reports (XML).

Le mécanisme de gestion des erreurs par exception est conservé dans le code du SIF car à ce niveau il ne semble pas devoir nuire aux performances