HAL
open science

# ESOPE : Une étape de la Recherche Française en Systèmes d'Exploitation (1968-1972)

Claude Bétourné, Jean Ferrié, Claude Kaiser, Sacha Krakowiak, Jacques Mossière

## To cite this version:

Claude Bétourné, Jean Ferrié, Claude Kaiser, Sacha Krakowiak, Jacques Mossière. ESOPE : Une étape de la Recherche Française en Systèmes d'Exploitation (1968-1972). 7ème Colloque sur l'Histoire de l'Informatique et des Télécommunications, ACONIT : Association pour un conservatoire de l'informatique et de la télématique, Nov 2004, Cesson-Rennes, France. pp.173-198. lirmm-00108869

**HAL Id: lirmm-00108869**

**https://hal-lirmm.ccsd.cnrs.fr/lirmm-00108869**

Submitted on 16 Nov 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# ÉSOPE : une étape de la recherche française en systèmes d'exploitation (1968-1972)

Claude Bétourné[1], Jean Ferrié[2], Claude Kaiser[3], Sacha Krakowiak[4] et Jacques Mossière[5]

1 *Université Paul Sabatier, IRIT*
`betourne@irit.fr`
2 *Université des Sciences et Techniques du Languedoc, LIRMM*
`ferrie@lirmm.fr`
3 *Conservatoire National des Arts et Métiers, Cedric*
`kaiser@cnam.fr`
4 *Université Joseph Fourier, IMAG-LSR et INRIA*
`sacha.krakowiak@imag.fr`
5 *Institut National Polytechnique de Grenoble, IMAG-LSR et INRIA*
`jacques.mossiere@imag.fr`

**Résumé.** Cet article retrace l'histoire d'ÉSOPE, un projet de système d'exploitation en temps partagé, mené à l'IRIA entre 1968 et 1972. Il présente la place d'ÉSOPE dans le contexte national et international de l'époque, résume et commente les principaux choix qui ont guidé sa conception et sa réalisation, et propose une analyse critique des aspects scientifiques du projet, de sa gestion, et de ses retombées.

**Abstract.** *This paper is a historical account of the* ÉSOPE *project, an experimental time-sharing system developed at IRIA between 1968 and 1972. The following aspects are presented: the situation of* ÉSOPE *in the national and international context of the period; a summary and comment of its main design and implementation decisions; and a critical assessment of the scientific and management aspects of the project, and of its follow-ups.*

## 1 Origines et contexte initial

Le projet ÉSOPE a été lancé en mi-1968, dans une période marquée par une importante transformation du paysage informatique français et par une intense activité de recherche dans le domaine des systèmes d'exploitation. L'orientation du projet a été influencée par ces deux aspects de son contexte initial, que nous rappelons brièvement avant de développer ses motivations et ses objectifs.

## 1.1 Contexte scientifique

La période 1966-68 est une phase cruciale de l'histoire des systèmes d'exploitation. On peut dire qu'en l'espace de 2 ou 3 ans ce domaine passe d'un stade de savoir-faire technique (d'ailleurs fort développé) à celui de discipline scientifique. Pour la première fois, une conférence patronnée par l'ACM s'intitule en 1967 *Symposium on Operating Systems Principles* ; ce sera la première édition de la série SOSP, aujourd'hui conférence bisannuelle de grand prestige dont la vingtième édition aura lieu en 2005. Le numéro de mai 1968 des *Communications of the ACM* reprend les meilleures communications de ce symposium. Il contient entre autres l'article de Dijkstra sur le système THE [26], qui aura une influence profonde et durable.

Au cours de ces années auront été élaborés les principes de base de la synchronisation (notion de processus, mécanisme des sémaphores), de l'adressage (segmentation), de la protection (première ébauche des capacités), des systèmes de fichiers hiérarchiques. Les premières mémoires paginées sont apparues, et les déboires des premiers systèmes à mémoire virtuelle avec pagination à la demande (écroulement) commencent à être analysés. On connaît quelques prototypes de recherche : THE, déjà cité, à Eindhoven ; CTSS [19] au MIT, précurseur de Multics [20] en cours de développement ; Genie [39] à Berkeley, bientôt suivi de CAL/TSS, MTS à l'université de Michigan. Le domaine est donc foisonnant, et la mise en forme de ces connaissances nouvelles est elle-même une activité proche de la recherche (voir [22]).

La conception du système ÉSOPE a été influencée par tous ces travaux. Nous y revenons en détail dans la section 2.

Les systèmes industriels qui vont intégrer certains de ces progrès n'auront en revanche pas d'influence sur ÉSOPE : ils utilisent des architectures trop éloignées du Sigma 7 (notre machine, voir 2.1) comme le Burroughs B5500 ou ne sont pas encore dévoilés en 1968. IBM travaille sur CP/67, encore expérimental, qui introduit les concepts de machine virtuelle et d'hyperviseur, et qui sera le support du très populaire CMS, lui-même ancêtre de VM/370. BBN (Bolt Beranek and Newman Inc.) développe Tenex sur un DEC PDP-10 [12]. Bull, en France, prépare l'architecture HB 64 et le système GCOS 64 qui s'inspirent de Multics et qui seront annoncés en 1974. Les premiers Unix apparaissent en 1969 aux Bell Labs sur PDP-7 et PDP-9 ; sur PDP-11, Unix est opérationnel en 1971 dans une version va-et-vient *(swapping)* à la CTSS, mais nous ne devions le découvrir qu'en 1973, après la fin d'ÉSOPE (SOSP 1973, puis CACM [44]).

## 1.2 Contexte institutionnel et naissance du projet

Le plan calcul avait été lancé en fin 1966, avec l'objectif de doter la France des capacités industrielles lui permettant d'être autonome dans le domaine des ordinateurs moyens. Outre son volet industriel, la Compagnie Internationale pour l'Informatique (CII) issue de la fusion des sociétés CAE et SEA, le plan calcul comportait un volet scientifique, l'Institut de Recherche en Informatique et Automatique (IRIA), qui avait pour mission de développer la recherche, la formation, et la diffusion de la connaissance scientifique et technique dans ses domaines de compétence. Mis en place courant 1967, l'IRIA était organisé en six « Directions de Recherche » largement autonomes. La Direction « Structure et Programmation des Calculateurs », sous la conduite de l'ingénieur en chef du génie maritime Henri Boucher, commençait à recruter ses membres en début 1968.

Le projet initial de la Direction de Recherche était de doter l'IRIA d'un système de calcul interactif (on disait alors « conversationnel ») multi-utilisateurs. Il faut se souvenir que ce mode de travail était à l'époque relativement peu développé, et que les terminaux étaient essentiellement des télétypes ; les consoles graphiques étaient une ressource rare, coûteuse, difficile à programmer, et il était prévu que nous en recevions une. Le langage PL/1, considéré à l'époque comme « universel », avait été choisi comme langage unique d'interaction.

La mission étant ainsi fixée, il fallait définir un programme de travail. L'IRIA disposait en mi-1968 d'un ordinateur CAE 90-80, machine de conception ancienne, qui arrivait en bout de course. La machine qui nous était promise, la CII 10070, avait beaucoup plus d'attraits (voir 2.1) ; mais sa date de livraison était incertaine, et éloignée d'au moins un an (elle ne devait arriver en fait que 20 mois plus tard). Son environnement logiciel nous était également inconnu ; nous savions, sans plus de précisions, que la CII travaillait à un système multiprogrammé (mais non en temps partagé), MMP (futur Siris 7).

Les voies d'approche possibles étaient en gros les suivantes.

1. Prévoir de construire notre environnement de programmation sur le système développé par la CII. Nous avions peu d'informations techniques sur ce système, la CII ne semblant pas à l'époque soucieuse d'établir avec nous des relations autres que de fournisseur à client.
2. Nous associer à un autre projet développant un système d'exploitation sur la 10070. Un tel projet, appelé SAM [45], était mené au CERA (Centre d'Études et de Recherches en Automatique, dépendant de Sup'Aéro), sous la direction de Jean-Paul Rossiensky et Vincent Tixier, et avec la participation de Jean-Yves Leclerc.
3. Développer nous-mêmes, sur la 10070 « nue », le système d'exploitation répondant à nos besoins.

Les voies 1 et 2 avaient l'inconvénient de nous rendre dépendants de partenaires dont les objectifs étaient différents des nôtres, qui avaient leur propre calendrier, et dont la volonté de coopération était à nos yeux loin d'être manifeste. La solution 3 n'était pas sans risques ; après tout, même si les plus anciens d'entre nous avaient une expérience de conception et de réalisation en informatique (calcul scientifique pour Bétourné, systèmes temps réel pour Kaiser, acquisition et traitement de données pour Krakowiak), la conception de systèmes d'exploitation était pour nous un domaine nouveau. Mais cette nouveauté même, surtout dans le contexte scientifique qui a été décrit, et le défi que représentait cette construction *ex nihilo*, emportèrent vite la décision. L'équipe initiale (alors composée de Bétourné, Boulenger, Kaiser, Krakowiak et Mossière – Ferrié devait nous rejoindre en janvier 1969) présenta en octobre 1968 cette proposition à notre directeur Boucher, qui l'approuva sans réserves. Le projet reçut le nom d'ÉSOPE (Exploitation Simultanée d'un Ordinateur et de ses PEriphériques).

L'appui de notre directeur de recherche nous donnait le feu vert vis-à-vis de la direction de l'IRIA. Il n'en fut pas de même pour la Délégation à l'Informatique (l'organe de pilotage du plan calcul), qui voyait d'un mauvais œil l'existence de deux projets sur le même thème (celui du CERA et le nôtre) et nous incita vivement à fusionner. Mais cette vue technocratique des choses ignorait les réalités du terrain (les choix de conception de SAM étant déjà faits, l'intérêt scientifique de la collaboration était pour nous voisin de zéro). Une amorce de collaboration fut néanmoins tentée avec SAM en vue de définir un langage commun d'implémentation, dans la ligne de PL360 [55], langage d'assemblage évolué muni de constructions syntaxiques de haut niveau. Cette tentative échoua en raison de divergences sur la conception ; chacun des projets définit son propre langage (LP70 [46] pour SAM, LP10070 [13] pour ÉSOPE) et les contacts s'arrêtèrent là[1].

Quant aux relations avec la CII, elles ne furent pas celles qui auraient dû résulter du cadre institutionnel du plan calcul. L'une des missions de l'IRIA était en effet d'élaborer les connaissances et le savoir-faire qui devaient aider la CII à développer sa propre compétence. Mais la CII ne semblait pas considérer notre projet comme un interlocuteur valable, et ne lui manifesta qu'un intérêt poli et épisodique. La Délégation à l'Informatique, pour sa part, n'avait qu'une idée vague de ce qu'était un projet de recherche, et semblait s'attendre à ce que nous fournissions un produit clés en main. Nous revenons sur ces aspects dans les sections 3.1, 3.3 et 4.

1. Une version de SAM fut développée au centre de recherche de la CII, après le départ de ses concepteurs vers fin 1969 ou début 1970, mais ce travail n'eut pas de suites. Une activité de recherche autour de SAM se poursuivit au CERT à Toulouse, après le transfert de Sup'Aéro dans cette ville.

### 1.3 Objectifs et priorités du projet

Le projet avait un double objectif :

— développer un système d'exploitation à l'usage interne de l'IRIA. Ce système devait servir de base à un environnement de programmation interactif et partagé pour 3 classes de travaux : l'utilisation du PL/1 en mode conversationnel sur une vingtaine de terminaux, l'exécution d'un travail de fond (*batch*), et enfin la supervision de tâches en temps réel[2].
— mener des recherches sur les principes de conception et de réalisation des systèmes d'exploitation ; avant tout, développer la compétence de l'équipe dans ce domaine : nous partions de quasiment zéro …

Il était clairement précisé que le prototype construit avait un caractère expérimental ; il ne s'agissait en aucun cas d'un produit, que nous n'avions d'ailleurs pas les moyens de développer, notre équipe de base comprenant 5 personnes (l'effectif est temporairement monté jusqu'à 7 ou 8). Si cette taille réduite nous a amenés à limiter l'ampleur de la réalisation, elle a eu l'avantage de simplifier les problèmes d'organisation ; l'homogénéité de l'équipe a en outre facilité la participation de chacun aux choix et aux décisions.

Pour la conduite du travail, nous avons donc fixé explicitement ou implicitement des sous-objectifs et des priorités. Certains de ces choix ont été orientés par l'objectif de réalisation, d'autres par l'intérêt scientifique (voire l'effet de mode), d'autres enfin sont dus à notre ignorance. En particulier, nous manquions d'expérience concrète d'utilisation d'un système d'exploitation interactif et partagé, ce qui nous a conduits à sous-estimer deux aspects :

— le rôle central du système de gestion de fichiers (SGF) dans un système d'exploitation,
— l'importance des outils de base constituant un environnement interactif (notamment le *shell* qui devait contribuer plus tard au succès d'Unix).

Nous avons donc concentré notre effort sur les points suivants.

— Problèmes liés au temps partagé, notamment ordonnancement des processus, gestion des terminaux, gestion des tâches conversationnelles.
— Problèmes liés à l'utilisation de la mémoire virtuelle, à l'époque assez peu explorés.
— Problèmes d'allocation de mémoire, notamment ceux liés à la gestion de mémoire paginée et à la prévention de l'écroulement.

2. Ce dernier aspect ne fut en fait jamais développé, faute sans doute d'avoir trouvé un domaine d'application.

La gestion des fichiers et des entrées-sorties non interactives n'a été abordée que relativement tard dans le projet.

D'autres aspects importants n'ont pas été pris en compte :

— Administration du système (paramétrage, génération, sécurité, journaux, mesures).
— Portabilité des applications depuis ou vers d'autres systèmes.
— Portabilité du système lui-même. Nous n'avons notamment pas cherché à définir une interface basse indépendante du matériel, ni d'interfaces internes standard.

Ces aspects n'étaient pas strictement nécessaires au fonctionnement d'un prototype expérimental et nous les avons explicitement exclus. Ils étaient par ailleurs, à l'époque, considérés comme marginaux sur le plan de la recherche, situation qui n'a commencé à changer qu'au début des années 1990.

Pour répondre à l'objectif de recherche et de formation, nous avons porté une grande attention à la formulation explicite des choix de conception, et à l'élaboration d'une documentation de conception préalable au codage[3]. Chacun des membres de l'équipe devait participer à la conception et au codage ; chacun devait avoir une connaissance à peu près complète de l'ensemble du code ; l'organisation était souple et les sous-groupes redéfinis en fonction des tâches. Ces principes ont été explicitement formulés au début du projet et ont en fait été à peu près suivis.

La suite de cet article développe trois vues du projet ÉSOPE. La section 2 présente une vue architecturale du projet « achevé », replacée dans le contexte de l'époque. La section 3 présente une vue chronologique du développement d'ÉSOPE. La section 4 conclut par une analyse critique du projet et de ses retombées.

## 2 Concepts et architecture du système ÉSOPE

Nous présentons dans cette section les principaux choix de conception et de réalisation d'ÉSOPE, en mettant en évidence les sources initiales d'inspiration, et également celles apparues au cours du déroulement du projet. Nous mettons également ÉSOPE en perspective par rapport à deux systèmes :

— Unix, aujourd'hui une référence centrale, bien que ce système soit resté inconnu de nous jusqu'en 1973 [44] et que la pagination à la demande n'y ait été introduite qu'en 1978 avec Unix BSD.
— VAX-VMS de DEC (descendant lointain de Tenex), l'un des meilleurs systèmes d'exploitation des années 1980.

3. L'important délai de livraison de la machine a été à cet égard un facteur bénéfique.

Un projet de système d'exploitation est nécessairement influencé par l'architecture de la machine sous-jacente, qu'il s'agisse d'exploiter au mieux ses capacités ou de s'accommoder de ses limitations. C'est pourquoi nous présentons brièvement les principaux traits de la CII 10070.

## 2.1 La machine CII 10070

La 10070 était la machine Sigma 7 [17] de la compagnie américaine Scientific Data Systems (SDS[4]), initialement distribuée sous licence, puis produite, par la CII. Annoncée en 1966 (et livrée la même année), la Sigma 7 était la première machine à mots de 32 bits succédant à une série de machines à 24 bits, comprenant notamment la SDS 940 (support du projet Genie à Berkeley), explicitement dédiée au temps partagé, mode d'utilisation très nouveau à l'époque. La Sigma 7 était annoncée comme ordinateur universel, pour des applications scientifiques, de gestion, et temps réel.

La Sigma 7 était de fait une excellente machine pour le temps réel, grâce à un système d'interruptions bien conçu, comprenant 237 niveaux hiérarchisés, pouvant individuellement être armés et masqués, ainsi qu'à une horloge temps réel avec comptage programmable. Ses performances étaient bonnes pour l'époque (voir ci-après). Sa vocation « temps partagé » se traduisait par la présence de jeux multiples de registres (jusqu'à 32 jeux) pouvant être permutés lors de la commutation du mot d'état[5], et par une mémoire virtuelle paginée « plate » (la correspondance entre pages virtuelles et pages physiques étant assurée par des registres topographiques permettant une traduction efficace et une commutation rapide entre espaces virtuels). Néanmoins, ce système d'adressage avait deux limitations importantes.

— L'adresse virtuelle (de mot) avait une taille de 17 bits (256 pages de 512 mots), comme l'adresse physique. De ce fait, la taille de l'espace virtuel était limitée à la taille maximale de la mémoire physique.
— Il n'y avait aucun mécanisme de translation d'adresse (registre de base ou autre). De ce fait, un adressage segmenté ne pouvait être que simulé par un mécanisme logiciel ad hoc.

Ces caractéristiques ont directement influencé la conception de la mémoire virtuelle d'ÉSOPE.

La configuration installée à l'IRIA (voir figure 4 en fin d'article) comportait :

— une mémoire centrale en tores de ferrite de 512 Koctets, de temps de cycle $1\mu s$,

4. En mars 1969, SDS fut absorbée par Xerox, donnant naissance à la compagnie Xerox Data Systems (XDS). Cette première tentative d'entrée de Xerox dans le domaine informatique devait tourner court en 1975.

5. Ce mécanisme, qui était en option, ne fut en fait pas utilisé par ÉSOPE.

— trois disques à têtes fixes de 3 Moctets chacun, avec une durée de rotation de 34 ms par tour et un temps de transfert de 13 ms par page[6],
— des périphériques classiques : lecteur de cartes, imprimante et dérouleurs de bandes,
— une mémoire de masse (DIMAS) de 200 Moctets équipée de disques à tête mobile,
— 24 téléimprimeurs SAGEM 8/200 à 20 caractères par seconde,
— une console graphique SINTRA VU-2000.

Un système d'exploitation en temps partagé pour la Sigma 7, UTS, avait été annoncé en 1966. Ce système ne fut en fait disponible qu'en 1971 et n'eut aucune influence sur ÉSOPE.

## 2.2 Architecture d'ensemble du système

L'organisation du système repose sur un nombre réduit de concepts, exploités à tous les niveaux de façon à obtenir une architecture claire et uniforme [6, 7].

Les activités du moniteur et des utilisateurs sont représentées par des processus séquentiels, à l'instar du système THE. On peut distinguer diverses familles de processus, les membres d'une même famille se partageant des données. Dans ce but est introduit le concept d'*usager*, comme association d'une mémoire virtuelle et d'un ensemble de processus s'exécutant dans cette mémoire. Chaque utilisateur interactif du système est ainsi représenté par un usager. Le partage de code réentrant entre usagers est possible, à travers le système de fichiers. La notion d'usager correspond à celle de processus dans Unix, les processus d'ÉSOPE étant en fait des *threads*.

Le partage des ressources principales, à savoir mémoire centrale et processeur(s), s'effectue entre usagers et non entre processus, pour des raisons indiquées en 2.5. Le système d'exploitation proprement dit (moniteur), qui assure la gestion des usagers, est un ensemble de processus cycliques, généralement résidant et s'exécutant en mémoire physique. Du point de vue de l'allocation de ressources, l'ensemble de ces processus est considéré comme un usager unique prioritaire sur les autres. La décomposition en processus est dirigée par des considérations d'indépendance logique [8].

Un autre choix de base d'ÉSOPE, guidé par des considérations d'uniformité, est celui d'un mécanisme unique de synchronisation entre processus, le sémaphore. En

6. Une anecdote amusante à ce propos : la vitesse de rotation des disques, donc leur débit, était asservie à la fréquence de l'alimentation électrique ; nous avons donc acheté un convertisseur de fréquence de 50 à 60 Hz pour obtenir les performances annoncées par le constructeur américain.

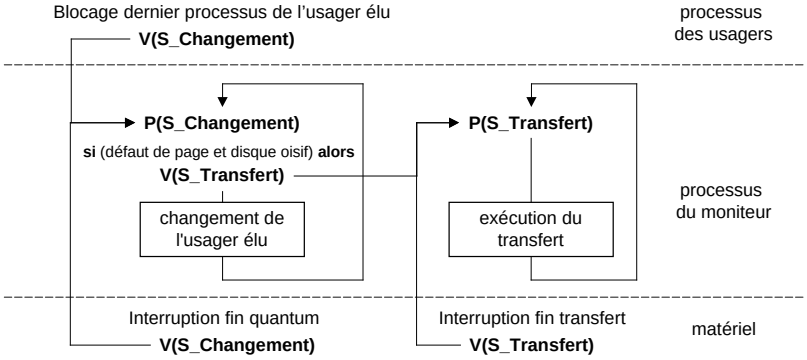


FIGURE 1 – Vue partielle de la coopération entre processus dans ÉSOPE.

conséquence, tous les événements asynchrones, qu'ils proviennent de la détection d'une condition logique ou d'une interruption, sont traduits par une primitive *V* qui réveille un processus chargé de traiter l'événement. Ainsi la figure 1 représente deux processus du moniteur et les événements qui les activent. L'un de ces processus gère le flux des usagers en mémoire et élit un usager pour l'attribution du ou des processeurs ; l'autre exécute les requêtes de transfert entre mémoire et disque.

Une partie des actions du moniteur s'exécute en espace virtuel ; en particulier, chaque usager possède un *processus premier*, représentant du moniteur, dont le rôle est précisé en 2.3. Les pilotes des entrées-sorties lentes constituent un usager particulier (voir 2.4.3), le pilotage des terminaux étant intégré au programme de gestion de chaque usager.

Indiquons enfin quelques éléments sur la taille du système. Dans sa dernière version, le code source d'ÉSOPE comprenait une quinzaine de modules totalisant 17 500 lignes ; la taille du noyau résidant était de 112 Koctets (56 pages), soit 22% de la mémoire physique ; la partie du système s'exécutant dans l'espace virtuel de chaque usager occupait 38 Koctets (19 pages). Aux normes actuelles, ÉSOPE serait plutôt un nano-noyau …

## 2.3 Gestion et synchronisation des processus

La notion de processus, en tant qu'activité séquentielle d'exécution d'un programme, était récente à l'époque. Outre l'article fondateur de Dijkstra [25], nos références dans ce domaine ont été les travaux de Saltzer au MIT [48] et de Lampson à Berkeley [38], qui avaient construit des noyaux de multiprogrammation et examiné les problèmes d'ordonnancement et la représentation des structures de données.

Le noyau développé dans ÉSOPE [27, 35] fournissait des primitives permettant à un processus de réaliser :

— des actions directes sur un processus, comme créer ou détruire un processus du même usager ;
— des interactions indirectes de synchronisation par sémaphores ;
— des actions sur son propre comportement, comme suspendre son exécution (pendant un délai), attendre (une date) ou s'autodétruire ;
— des accès à certaines données du système, aux fichiers et aux segments des processus ou des usagers, en particulier par couplage (voir 2.4.2).

Chaque processus est contrôlé par un *pouvoir* qui définit ses droits tant sur l'exécution de ces actions que sur l'utilisation des ressources physiques (mode – maître ou esclave – d'utilisation de l'unité centrale, clés de protection câblée des pages mémoire, masquage de certaines interruptions, etc.). Pour cela les processus sont divisés en 4 classes, chacune ayant un ensemble d'actions permises :

— les processus du moniteur résidant en mémoire,
— le processus premier de chaque usager qui a deux rôles principaux, la gestion de l'usager, (à ce titre, il a des droits d'accès aux sémaphores du moniteur et peut interrompre ou détruire tous les processus de l'usager), et l'interprétation du langage de commande,
— les traducteurs, fils du processus premier, qui utilisent du code partagé entre usagers,
— les processus d'un usager, créés pour un utilisateur du système, et qui ne peuvent directement interagir qu'avec les processus du même usager (par des noms locaux de sémaphores ou par la mémoire virtuelle partagée).

Le choix des sémaphores comme mécanisme de synchronisation, tant pour les processus du moniteur que pour les processus applicatifs, a conduit à associer deux mécanismes d'accès à une structure de données unique en mémoire. Les processus du moniteur utilisent un numéro interne et un accès direct à l'intérieur du noyau. Les processus utilisateurs connaissent les sémaphores par un nom local à chaque usager et la correspondance de ce nom avec le numéro interne est établie par le noyau à la suite d'un appel système.

Pour simplifier leur gestion, les processus et sémaphores du moniteur sont en nombre fixe alors que les processus et les sémaphores des usagers sont créés et détruits dynamiquement.

Ce mode de gestion s'est révélé trop figé, en particulier pour permettre la souplesse nécessaire à la génération et à l'extension du système lorsqu'il est devenu complexe. Il aurait sans doute fallu permettre une création dynamique des processus

et des sémaphores du système, fournir un système de communication entre usagers par boîtes aux lettres et donner plus de dynamisme à la gestion des droits des processus.

Le mécanisme uniforme de gestion des processus a permis une instrumentation simple du suivi de leur exécution concurrente, en notant dans un tampon circulaire la date et le mot d'état programme d'un processus à chaque commutation (qui ne pouvait être occasionnée que par une opération sur un sémaphore ou par la création ou la fin d'un processus). L'horloge temps réel a été utile pour comptabiliser le temps passé dans chaque processus.

Comme indiqué en 1.3 (note 2), l'usager temps réel initialement prévu n'a jamais été développé, mais tous les ingrédients existaient pour le faire : un noyau réentrant permettant une prise en compte rapide des interruptions (les primitives longues étaient interruptibles et découpées en sections accessibles en exclusion mutuelle) ; la gestion par priorité des files de processus et de sémaphores ; le « collage » de pages virtuelles en mémoire physique pour y fixer tout ou partie d'un processus (ces mécanismes furent plus tard présents dans VAX-VMS). Il manquait une interface d'utilisation, c'est-à-dire un jeu de coupleurs vers des capteurs ou actionneurs temps réel.

## 2.4 Gestion de l'information

Nous décrivons successivement la gestion des fichiers, la constitution de la mémoire virtuelle par couplage et le système de gestion des entrées-sorties.

### *2.4.1 Système de fichiers*

Pour les raisons indiquées en 1.3, nous n'avons pas porté un effort important sur la conception du système de gestion des fichiers. Nous nous sommes contentés de reprendre, en les simplifiant, les idées développées dans Multics, reprises ensuite dans Unix et qui sont toujours utilisées aujourd'hui. Un fichier d'ÉSOPE est un simple flot d'octets sans structure. Pour permettre le couplage en mémoire virtuelle (voir 2.4.2), il est divisé en blocs (articles) dont la taille est égale à celle d'une page. L'implantation des blocs d'un fichier est contenue dans une table d'implantation du fichier (TIF), jouant un rôle analogue à la table des *i-nodes* d'Unix.

Un système de catalogues hiérarchiques, limité à deux niveaux[7], permet la désignation symbolique des fichiers. Lorsqu'un fichier est ouvert, il est également désigné

7. Cette décision paraît aujourd'hui surprenante ; elle s'explique probablement par le fait que (dans un contexte de ressources limitées) l'effort supplémentaire de programmation nécessaire pour construire un système hiérarchique complet ne nous a pas paru prioritaire par rapport aux travaux sur l'allocation de ressources, jugés plus novateurs.

par un nom interne, entrée dans la table des fichiers ouverts (TFO). Cette entrée contient un pointeur vers la TIF du fichier en mémoire.

Nous n'avons prêté que peu d'attention à la permanence des informations. Toutes les tables et tous les catalogues résidaient en mémoire centrale. À la fin d'une session d'exploitation du système, l'ensemble de la mémoire centrale pouvait être vidé sur bande magnétique ; ce vidage constituait le seul moyen de sauvegarde.

### *2.4.2 Mémoire virtuelle et couplage*

Un espace virtuel d'ÉSOPE fournit un moyen de désigner un ensemble de blocs de fichier en y associant des adresses virtuelles. Une opération appelée *couplage* permet de mettre en correspondance une page avec un bloc de fichier.

Chaque espace virtuel est décrit par une table des pages virtuelles (TPV) qui fournit pour chaque page son adresse en mémoire physique (ou 0 si la page n'est pas en mémoire) et le bloc de fichier auquel elle est couplée (numéro de fichier ouvert et numéro de bloc dans le fichier).

La figure 2 présente l'ensemble des structures de données utilisées pour localiser les différentes pages. La partie a) illustre l'utilisation des espaces virtuels comme fenêtres d'accès à l'espace des fichiers ; la partie b) montre la localisation d'une information désignée par une adresse virtuelle.

Nos sources d'influence étaient là encore Multics et les articles de base sur la segmentation (Dennis [24], Arden et al. [2]), mais aussi un système prototype, Gordo [1], réalisé à l'UCLA sur une Sigma 7 et qui utilisait le couplage. Nous avons à l'époque sous-estimé le caractère fondateur de l'article de Dennis et étudié de façon détaillée l'article d'Arden et al., beaucoup plus proche de la réalisation.

Nous avons cherché à pallier l'absence de segmentation dans le 10070 en la simulant par le mécanisme du couplage. Mais les contraintes imposées par un espace virtuel linéaire subsistaient. En particulier, des pages contenant des programmes partagés, donc des adresses virtuelles absolues, devaient être couplées aux mêmes adresses virtuelles pour permettre un partage en mémoire centrale (cas du fichier *s1*, fig. 2 a).

De façon curieuse, nous n'avons pas songé à permettre le partage de code entre processus d'un même usager. Ce partage était réalisé entre usagers, et cela nous a paru suffisant. Partager le code à l'intérieur d'un usager aurait toutefois amené à imposer des conventions de programmation (accès indirect aux données) ou à modifier la mémoire virtuelle à chaque commutation de processus.

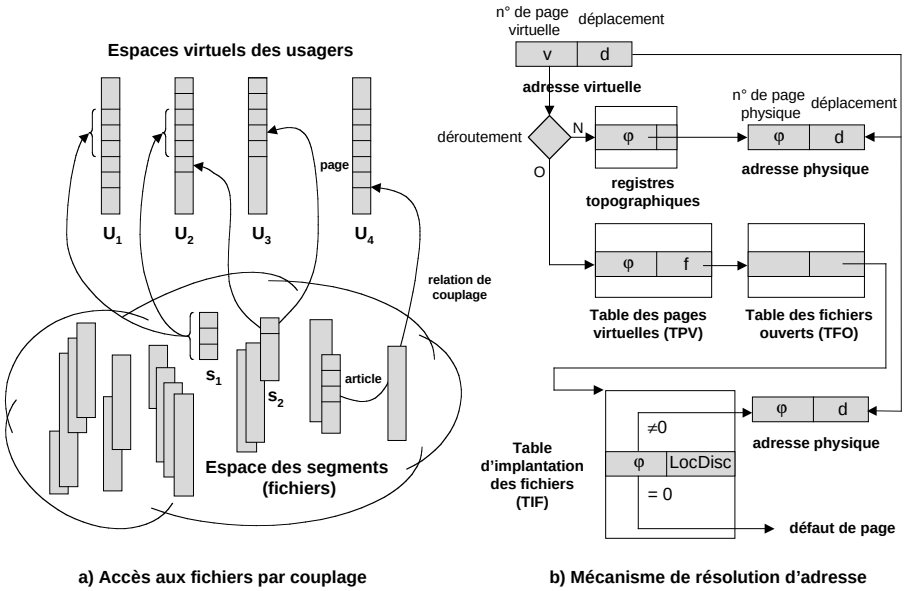


FIGURE 2 – Gestion de l'information dans ÉSOPE.

### *2.4.3 Entrées-sorties*

Les échanges avec les fichiers sur disque sont effectués uniquement, après couplage en mémoire virtuelle, via les mécanismes de gestion de mémoire (cf. 2.5). Le système d'entrées-sorties d'ÉSOPE [3] ne concerne que les échanges avec les périphériques lents : imprimante et lecteur de cartes.

Un usager particulier, l'usager PTT, reçoit via une boîte aux lettres gérée en producteur-consommateur des demandes d'entrée ou de sortie portant sur un fichier complet. À chaque périphérique sont associés deux processus.

— Le facteur du périphérique est responsable du couplage des différentes pages composant le fichier dans des pages virtuelles qui sont collées en mémoire physique jusqu'à la fin de leur transfert. Ce processus s'exécute en adressage virtuel.
— Le pilote du périphérique assure les transferts entre périphérique et mémoire ; pendant chaque exécution d'une entrée-sortie, le pilote est bloqué sur un sémaphore privé, le déblocage étant effectué par un *V* exécuté par le traitant de l'interruption d'entrée-sortie. Le pilote s'exécute en adressage physique.
— Le facteur et le pilote se synchronisent également selon le schéma producteur-consommateur.

Relu avec recul, le système d'entrées-sorties d'ÉSOPE reste un joli exercice de programmation de processus parallèles communiquant systématiquement par des boîtes aux lettres gérées selon le schéma producteur-consommateur. Un processus chien de garde permet de détecter des erreurs de périphériques ou des pertes d'interruptions. Des commandes permettent à l'opérateur d'immobiliser un périphérique pour permettre des opérations de maintenance (changement du papier de l'imprimante, par exemple).

### 2.5 Allocation de ressources

Nous avons choisi l'usager, plutôt que le processus, comme allocataire des ressources physiques, mémoire et processeur. Ce choix a résulté du souci de traiter globalement l'allocation de ces deux ressources (voir plus loin) ; par ailleurs, si un usager a plusieurs processus, le temps de commutation de la mémoire virtuelle (long par rapport à celui du processeur), est amorti sur l'ensemble de ces processus.

L'unité centrale est donc allouée en tourniquet aux différents usagers et réquisitionnée à l'expiration d'un quantum d'unité centrale. À l'intérieur d'un usager, l'UC est allouée au processus le plus prioritaire, qui devient le processus élu. Le passage de l'unité centrale d'un processus à un autre ne peut avoir lieu qu'en cas de blocage du processus élu ou d'activation d'un processus plus prioritaire du même usager.

Seuls les usagers prêts (ayant au moins un processus activable) sont candidats à l'admission en mémoire centrale, puis à l'exécution de leurs processus. À chaque usager est associée une catégorie, estimation du nombre de pages nécessaires à son exécution efficace. Un usager n'est admis en mémoire que si le système peut lui réserver un nombre de cases égal à sa catégorie. Les catégories sont modifiées dynamiquement en fonction du comportement individuel de chaque usager. La réservation de mémoire est effectuée pour un intervalle de temps maximum appelé temps de résidence. Un usager est également chassé de la mémoire en cas de dépassement de sa catégorie ; il y sera réadmis avec une catégorie supérieure.

Le chargement des pages est effectué à la demande et l'algorithme de remplacement est un algorithme FIFO par usager, imposé par l'absence de bit d'utilisation des pages. Pour diminuer le nombre des transferts de page, plusieurs améliorations ont été implantées (distinction entre pages modifiées et intactes, et récupération des pages en mémoire qui avaient été libérées, mais non encore recopiées sur disque). Pour réduire la durée des transferts en réorganisant l'ordre des demandes pour minimiser le nombre de tours de disque nécessaire, les demandes de pages sont regroupées pour tous les processus d'un usager et sont acquittées globalement.

Enfin, les programmes partagés sont gérés de façon à permettre le partage en mémoire physique de leurs cases. L'économie était substantielle dans la mesure où

nous disposions d'une seule vraie application, le compilateur-interprète CPL/1 qui était exécuté par l'ensemble des usagers.

Une simulation[8] de l'allocation de ressources [9] permit de valider les principaux choix (gestion des usagers, partage, abandon du préchargement dont l'intérêt était marginal).

ÉSOPE est l'un des premiers systèmes à avoir géré globalement la mémoire et l'unité centrale. Sa gestion des ressources a bénéficié des nombreuses publications des années 1968-70 traitant du comportement des programmes en mémoire paginée et de régulation de charge. Les principales inspirations viennent des travaux effectués à IBM ([37] pour la pagination, [16, 51] pour la régulation de charge), au MIT (Denning et le modèle du working set [23]) et à Edimbourg (nous avions suivi un cours de Whitfield décrivant le système EMAS—Edinburgh Multi Access System [54, 50] et eu de nombreux échanges avec Shelness, autre membre de ce projet). Les résultats étaient satisfaisants : une quinzaine d'utilisateurs pouvaient travailler simultanément en interprétant des (petits) programmes PL1.

Notons pour conclure que la gestion de ressources du système VAX-VMS [41] était extrêmement voisine de celle adoptée pour ÉSOPE, ce qui valide nos choix a posteriori.

## 3 Historique du projet

Nous présentons les étapes les plus marquantes du projet, avant de traiter de deux aspects connexes : les activités externes liées au projet, et ses relations avec la CII.

### 3.1 Chronologie du projet

La figure 3 donne une vue synthétique du déroulement du projet, articulée autour des 5 versions successives du système. On y trouve également une vue des projets associés et des activités connexes (voir 3.2).

Une *version* correspond à un état supposé stable du système, qui marque une étape de la conception et de la réalisation, et constitue la base d'expérimentation jusqu'à la sortie de la version suivante. Chaque version avait pour but d'intégrer une fonction nouvelle importante, et de rendre dynamiques des aspects définis statiquement dans les versions antérieures. Les grandes lignes du contenu de chaque version sont indiquées dans la table 1. On notera que les versions 4 et 5 intègrent

8. Le programme de simulation, écrit en Simula, fut exploité en 1970 au Centre de Calcul Scientifique de l'Armement (CCSA), sur un Univac 1108 qui disposait d'un compilateur de ce langage.

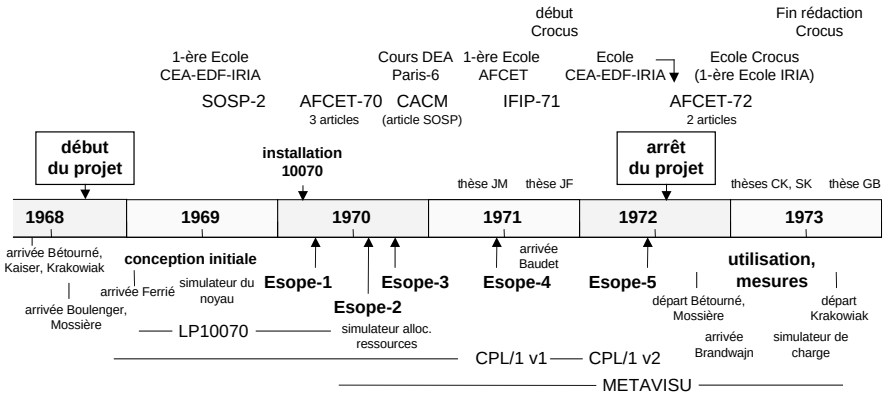


FIGURE 3 – Chronologie du projet ÉSOPE.

les systèmes interactifs réalisés au-dessus d'ÉSOPE : environnement CPL/1[11] et système graphique MÉTAVISU [40, 14].

L'examen de la figure 3 met en évidence les points suivants :

— La longueur de la phase initiale de conception (environ 18 mois). Cette phase (en partie imposée par le délai de livraison de la machine) nous a permis de réaliser une conception globale (les 6 premiers mois environ) puis une préparation détaillée de la mise en œuvre de chaque partie, documentée dans une dizaine de notes techniques ; la conception du noyau de gestion de processus a été validée par une simulation en PL/1. Notons que la version de l'article [6] présentée à SOSP en 1969 reposait essentiellement sur le travail de conception.
— Le délai relativement court entre les versions successives. Ce point est certainement une conséquence du précédent, et de l'utilisation du langage LP10070 de préférence au langage d'assemblage (sauf pour quelques parties restreintes du pilote des disques et du programme d'amorçage). Nous n'avons pas noté de problème majeur de développement, bien que nos outils de mise au point aient été rudimentaires.
— L'importance des activités collatérales liées à la formation. Ce point est développé en 3.2.
— La brutalité de l'arrêt du projet. Ce point est développé ci-après.

La version ÉSOPE-5 (juin 1972) permettait d'utiliser le langage PL/1 en mode conversationnel sur 16 terminaux avec un temps de réponse acceptable ; cette version servit de support à plusieurs séances de travaux pratiques pour des cours d'analyse numérique. Néanmoins, à partir du début de 1972, la situation du projet était devenue politiquement inconfortable : c'était un projet de recherche (lar-

| version | date | caractéristiques |
|---|---|---|
| ÉSOPE-1 | février 1970 | Décomposition en processus, gestion de 2 usagers interactifs avec mémoires virtuelles préparées « à la main », application minimale (calculette) |
| ÉSOPE-2 | juillet 1970 | Adressage virtuel, module de gestion du disque, usagers statiquement définis |
| ÉSOPE-3 | octobre 1970 | Gestion dynamique des processus, allocation de ressources statique, pas de fichiers |
| ÉSOPE-4 | juin 1971 | Gestion dynamique des ressources, système de fichiers, CPL/1 v. 1 |
| ÉSOPE-5 | juin 1972 | Entrées-sorties, langage de commande, utilitaires divers (chargeur, système de sauvegarde, éditeur de textes), système de mesure, CPL/1 v. 2, système graphique MÉTAVISU |

TABLE 1 – Les versions d'ÉSOPE.

gement reconnu au plan international), mais non évalué comme tel (l'IRIA n'avait aucun processus formel d'évaluation, et les membres du projet aucun statut académique) ; et d'autre part, les responsables de la CII et la Délégation à l'Informatique ne raisonnaient qu'en termes d'utilité immédiate et méconnaissaient le potentiel de compétence et de savoir-faire du projet. Le changement de direction qui eut lieu à l'IRIA en juillet 1972 donna l'occasion de faire un sort à cette activité devenue encombrante en décidant unilatéralement son arrêt. Notre proposition de livrer le système au centre de calcul de l'IRIA pour une utilisation expérimentale ne fut même pas sérieusement examinée.

Après le départ en octobre 1972 de Bétourné et de Mossière, une certaine activité se poursuivit néanmoins pendant un an autour d'ÉSOPE-5. L'utilisation du système pour des séances de TP permit de réaliser quelques mesures [10]. Un simulateur de charge fut réalisé par Alexandre Brandwajn, qui fit un bref séjour auprès d'ÉSOPE avant de poursuivre des travaux sur la modélisation des systèmes [15]. Le système servit également de support au travail sur CPL/1 et MÉTAVISU. Chaque équipe avait besoin d'espaces d'adressage de grande taille et utilisait le système en *stand-alone*. Le système fut ainsi utilisé assez longtemps après la dispersion de notre équipe, sans difficulté majeure.

### 3.2 Activités connexes

Nous avons eu la chance de participer en 1969 à la première École d'été d'informatique CEA-EDF-IRIA, qui avait pour thème les systèmes d'exploitation, et qui nous a permis de fructueuses interactions avec Dijkstra, Randell et Whitfield dans

la phase de conception d'ÉSOPE. Nous fûmes ensuite nous-mêmes sollicités pour transmettre notre expérience : d'abord en décembre 1970 pour une formation destinée aux étudiants de DEA et aux chercheurs de Paris-6 ; puis pour un cours à la première École d'été de l'AFCET en juillet 1971. Ce cours fut à l'origine de la création du groupe Crocus [22] qui devait marquer l'enseignement des systèmes en France pour une dizaine d'années. La première École organisée par Crocus (Les Arcs, mars 1973) fut également la première École de l'IRIA. L'École CEA-EDF-IRIA de 1972 (avec Coffman, Pinkerton et Westcott) fut consacrée aux modèles et mesures de systèmes, thème en phase avec nos préoccupations du moment.

Après la fin du projet, nous avons organisé à l'IRIA en avril 1974 un symposium sur les systèmes d'exploitation [29], qui réunit près de 200 personnes avec une très large participation internationale. Le comité de programme, présidé par Arden, comprenait notamment Habermann et Randell.

### 3.3 Relations avec la CII

Si les relations institutionnelles restèrent toujours difficiles avec la CII, qui refusa toute collaboration pendant le projet, elles furent souvent bonnes, à titre personnel, avec des concepteurs de systèmes à la CII pendant et après le projet. Mais leur impact resta très limité.

Avec la CII, Kaiser eut des contacts avec Alan Woodcock lorsque celui-ci fut chargé, pour la conception du système Siris 8 de l'IRIS 80, de dépasser les limitations d'adressage du système Siris 7 de la 10070 qui ne pouvait pas gérer des programmes de plus de 128 Kmots de mémoire virtuelle. Ce fut l'occasion de lui expliquer nos choix de gestion de l'adressage. Encore avec la CII à titre personnel, les travaux sur ÉSOPE et sur les architectures à domaines ont incité Denis Derville, René Chevance et Jean-Louis Mansion à demander en 1974 à Kaiser de participer au projet Y de conception d'une nouvelle architecture [18].

À la Bull[9], avec Jean Bellec, André Bensoussan, Claude Carré, Dave Slosberg (ceux des concepteurs du HB 64 que nous avons rencontrés dans des congrès), les rapports furent excellents, car ils furent les seuls interlocuteurs français à comprendre finement notre démarche architecturale. Mais ces contacts restèrent très limités car le secret industriel les contraignait. L'*interior decor* du HB 64 avait déjà été figé ; dès 1968 et 1970, la segmentation, les sémaphores câblés, la notion de groupe de *threads* avaient été retenus par eux et ils avaient trouvé dans nos choix un encouragement à poursuivre dans cette voie (notre équipe ne le savait pas, et cela nous aurait bien sûr confortés dans nos propres choix). Ce fut Bensoussan qui nous faci-

9. Bull (Bull-General Electric jusqu'en septembre 1970, Honeywell-Bull ensuite) était alors une société distincte de la CII, avec laquelle elle devait fusionner en 1975.

lita une visite auprès de l'équipe de Multics en 1969 et accueillit Kaiser et Krakowiak au MIT lors de cette visite.

## 4 Analyse critique

Nous présentons une analyse critique du projet, sous ses aspects scientifiques et politiques, avant de conclure sur son bilan final.

### 4.1 Aspects scientifiques

ÉSOPE n'a pas introduit d'idée vraiment révolutionnaire, mais (dans le courant lancé par THE) a montré qu'un système d'exploitation en temps partagé pouvait être organisé selon un modèle rigoureux de décomposition et de structuration. Il a raisonnablement couvert les principaux aspects d'un système réel (moyennant les réserves exprimées ci-après), ce qui n'est pas si fréquent dans un projet de type académique. Il a bien intégré un ensemble de techniques à la pointe de l'état de l'art de l'époque, tant pour la synchronisation des processus que pour la gestion de l'information et l'allocation de ressources.

Le système a servi de support à des applications de complexité substantielle (compilateur-interprète PL/1 et système graphique) et leur a apporté l'environnement nécessaire à leur fonctionnement. Malgré une période restreinte d'utilisation effective (due à son arrêt prématuré), il a fait l'objet d'une évaluation soigneuse quantitative (simulation [9] et mesures [10]) et qualitative (analyse rétrospective [34]). Une intéressante analyse[10] de quelques pannes résiduelles particulièrement subtiles (dont l'une était due à un dysfonctionnement d'une instruction de la machine) figure dans [33].

Le recul du temps montre quelques erreurs manifestes. Nous avons sous-estimé la place centrale du système de gestion de fichiers tant pour les usagers que pour la gestion du système lui-même (permanence des structures de données internes) ; le support nécessaire existait (suite d'octets accessible par blocs au moyen du couplage) mais a été mal exploité. Nous n'avons pas fourni de gestion organisée de l'espace de travail des processus (pile), laissant cette tâche à la charge de chaque application. Nous n'avons prêté que peu d'attention aux aspects d'administration, mais cette lacune semble avoir été partagée par nombre de prototypes de recherche de l'époque. Enfin, (faute sans doute d'avoir nous mêmes utilisé réellement le système) nous n'avons pas fourni aux utilisateurs l'environnement courant nécessaire

10. C'est ce travail qui a incité Flaviu Cristian, qui commençait alors sa thèse à Grenoble, à s'intéresser à la tolérance aux fautes. Il a ensuite mené dans ce domaine la carrière que l'on connaît, interrompue par sa disparition prématurée en 1999.

au travail interactif : nous n'avons aucun équivalent du *shell* Unix et de ses outils devenus standard.

La notion même de génie logiciel était naissante à l'époque : rappelons que le terme de *software engineering* a été créé en 1968. Nous avons néanmoins défini et appliqué de bonnes pratiques dans ce domaine. La conception du système a été très abondamment documentée : près d'une trentaine de rapports internes, sans compter les très nombreuses (environ 70) notes informelles[11] (parfois fort substantielles) échangées sur des aspects techniques.

La définition des versions successives s'est directement appuyée sur la structuration du système, et a bien fonctionné : ainsi, l'ensemble des processus du moniteur étaient présents dès la première version ; leur contenu était rudimentaire mais leur mécanisme d'interaction était en place et a donc pu être validé très tôt.

Nous avons mis en pratique avant la lettre les principes de l'*egoless programming* [53] : chacun avait une connaissance suffisante de l'ensemble du code pour pouvoir intervenir dans tout module. Nous avons néanmoins sous-estimé l'intérêt de construire nos propres outils de mise au point, notamment en mémoire virtuelle. Le seul outil réalisé (le journal d'exécution des processus) s'est révélé très précieux. Nous avons aussi découvert à l'usage les erreurs subtiles résultant de l'absence de contrôle de types entre modules[12].

Le projet a bien joué son rôle de formation, comme en témoignent les thèses qui en sont issues de manière directe ou indirecte [35, 36, 43, 27, 4, 15], l'implication dans de nombreuses Écoles dont beaucoup ont été des « premières » (voir 3.2) et la création du groupe Crocus.

Le projet a été bien reconnu au plan international (SOSP et CACM [6], IFIP [8, 34]). Plus important encore, il a été plusieurs fois cité dans les ouvrages de référence des années fin 70 [49, 42, 28] comme modèle de bonne structuration ; Habermann [32] le cite parmi *several systems that made history*, en compagnie de Multics, THE et Tenex.

### 4.2 Aspects politiques

Le projet a à la fois bénéficié et souffert des conditions particulières qui prévalaient à l'IRIA, institut nouvellement créé dans le cadre du plan calcul.

— il a bénéficié de la grande autonomie initiale des directions scientifiques ; notre directeur Henri Boucher nous a fait confiance a priori, et nous avons

11. Dont l'appellation « fusées volantes » dans notre jargon de l'époque traduisait le caractère spontané.

12. Ce qui a plus tard motivé certains d'entre nous à travailler sur les environnements de programmation pour systèmes composables.

eu pratiquement carte blanche pendant 3 ans, avec peu de contraintes administratives et surtout avec la possibilité de nous consacrer au projet à plein temps.

— il a en revanche cruellement souffert de l'absence (à l'époque) d'une réelle culture de l'évaluation scientifique à l'IRIA. La décision d'arrêter le projet a été prise sans consultation et sans qu'une expertise sérieuse ait été menée.

— il a enfin souffert du flou de la doctrine de l'époque sur les objectifs et les modalités du transfert de résultats de recherche vers l'industrie (en l'occurrence l'industriel « obligé », la CII). L'arrêt du projet a introduit les premiers doutes sur la capacité de l'IRIA à réaliser un tel transfert. Il faudra attendre (longtemps) l'invention des *start-up* de l'INRIA pour trouver une formule qui fonctionne.

L'arrêt prématuré du projet, suivi de la dispersion de l'équipe, a contribué pour sa part à la régression de la place de la recherche française en systèmes, régression qui a été parachevée par la politique inepte des années 1974-80 (gel des postes et même des bourses doctorales, blocage des acquisitions de matériel[13]), dont nous payons encore le prix aujourd'hui.

Malgré cela, l'existence d'un projet comme ÉSOPE a sans doute encouragé et conforté le développement de recherches sur les systèmes en France, notamment à Rennes (projet SAR [52]) et Grenoble (projet GEMAU [31]). Mais il ne faut pas oublier qu'il existait déjà, avant ÉSOPE, des travaux au centre scientifique IBM de Grenoble [5] et à l'Institut de programmation à Paris [30].

### 4.3 Bilan final

Les plus importantes retombées d'ÉSOPE restent l'élaboration des connaissances et la formation. Retombée directe, car les membres du projet ont ensuite essaimé dans divers centres universitaires où ils ont contribué à créer ou à renforcer les activités de recherche et d'enseignement en systèmes ; retombée indirecte, notamment au travers des groupes Crocus [22] et Cornafion [21], où les membres du projet ont tenu une place importante. Le soutien initial apporté aux équipes qui gravitaient autour d'ÉSOPE (graphique, compilation) a également été appréciable. Enfin, l'activité de recherche en systèmes poursuivie à Rocquencourt après l'arrêt du projet (mais sous l'impulsion de ceux de ses membres restés sur place) a conduit à terme à la création du projet Chorus [47] et à ses propres retombées scientifiques et industrielles.

13. Il faut se souvenir que les restrictions sur l'achat de matériel informatique par les organismes publics ont *de facto* interdit, pendant plusieurs années, tout accès au système Unix à la communauté de recherche française.

FIGURE 4 – La CII 10070 de l'IRIA (vers 1973)
De gauche à droite : Kaiser, Lemaire, Saltel, X . . . , Boulenger (devant la console graphique), Gros

**Sources et remerciements** Les auteurs de cet article sont les membres de l'équipe qui a conçu et réalisé le système ÉSOPE. Ils tiennent à remercier les personnes suivantes qui ont été, à un moment ou à un autre, associées à la vie du projet : le directeur de recherche, Henri Boucher ; les réalisateurs du compilateur LP10070 : Jacques Boulenger, Jean-Yves Babonneau, Bui Tuong Phong (†), Jean-Jacques Lévy ; Gérard Baudet, qui a contribué au système d'entrées-sorties ; les membres de l'équipe CPL/1 : France Blaizot, Laurent Blaizot, Bernard Lorho, Michel Vatoux ; les membres de l'équipe MÉTAVISU : Pierre Boullier, Jacques Gros, Pierre Jancène, Alain Lemaire, Francis Prusker, Éric Saltel ; Marc Kronental, qui a étendu LP10070 pour une application de jeu de simulation ; Jean Kott, qui a participé aux travaux initiaux de conception ; Alexandre Brandwajn, qui a contribué à l'évaluation quantitative du système.

## Références bibliographiques

[1] G. B. ANDERSON, K. R. BERTRAN, R.W. CONN, K.O. MALMQUIST, R.E. MILLSTEIN & S. TOKUBO, « Design of a Time-Sharing System Allowing Interactive Graphics », *in Proc. ACM National Conference*, p. 1–6 (1968).

[2] B. W. ARDEN, B. A. GALLER, T. C. O'BRIEN & F. H. WESTERVELT, « Program and Addressing Structure in a Time-Sharing Environment », *Journal of the ACM*, vol. 13 (1), (1966) p. 1–16.

[3] G. BAUDET, J. FERRIÉ, C. KAISER & J. MOSSIÈRE, « Entrées-sorties dans un système à mémoire virtuelle », *in Actes du Congrès AFCET* (Grenoble, 1972).

[4] Gérard BAUDET, *Étude du comportement d'un tambour soumis à des demandes de transfert groupées*, Thèse de 3e cycle, Université Pierre et Marie Curie, Paris-6 (1973).

[5] Jacques BELLINO, *Mécanismes de base dans les systèmes superviseurs : conception et réalisation d'un système à accès multiples*, Thèse de doctorat ès sciences appliquées, Université de Grenoble (1973).

[6] C. BÉTOURNÉ, J. BOULENGER, J. FERRIÉ, C. KAISER, S. KRAKOWIAK & J. MOSSIÈRE, « Process Management and Resource Sharing in the Multiaccess System Esope », *Communications of the ACM*, vol. 13 (12), (1970) p. 727–733, [Version étendue d'une communication présentée au *Second ACM Symposium on Operating Systems Principles*, Princeton (NJ, USA), oct. 1969].

[7] C. BÉTOURNÉ, J. BOULENGER, J. FERRIÉ, C. KAISER, S. KRAKOWIAK & J. MOSSIÈRE, « Trois articles : Présentation générale du système Ésope ; Espace virtuel dans le système Ésope ; Allocation de ressources dans le système Ésope », *in Actes du Congrès AFCET* (1970).

[8] C. BÉTOURNÉ, J. FERRIÉ, C. KAISER, S. KRAKOWIAK & J. MOSSIÈRE, « System Design and Implementation using Parallel Processes », *in* W. H. FREEMAN, , *Information Processing, Proceedings of IFIP Congress 71*, TA-3, p. 31–36 (North-Holland, 1972, Ljubljana, 1971).

[9] C. BÉTOURNÉ & S. KRAKOWIAK, « Simulation de l'allocation de ressources dans un système conversationnel à mémoire virtuelle paginée », *Revue RAIRO-Informatique*, vol. B-1, (1974) p. 5–10, [Version étendue d'une communication présentée au Congrès AFCET, Grenoble, nov. 1972].

[10] C. BÉTOURNÉ & S. KRAKOWIAK, « Mesures sur un système conversationnel », *Revue RAIRO-Informatique*, vol. B-2, (1975) p. 5–15.

[11] F. BLAIZOT, L. BLAIZOT, B. LORHO & M. VATOUX, « Organisation du compilateur interpréteur CPL/1 », *in Actes du Congrès AFCET* (Paris, 1970).

[12] Daniel G. BOBROW, Jerry D. BURCHFIEL, Daniel L. MURPHY & Raymond S. TOMLINSON, « TENEX, a Paged Time Sharing System for the PDP-10 », *Communications of the ACM*, vol. 15 (3), (1972) p. 135–143, [Version étendue d'une communication présentée au *Third ACM Symposium on Operating Systems Principles*, Stanford University (CA, USA), oct. 1971].

[13] J. BOULENGER & M. KRONENTAL, « An Experience in System Programming Using a PL360-like Language », *in Proceedings of the IFIP WG-2.4 Meeting* (La Grande Motte, 1974).

[14] P. BOULLIER, J. GROS, P. JANCÈNE, A. LEMAIRE, F. PRUSKER & É. SALTEL, « MÉTAVISU : A General Purpose Graphic System », *in IFIP Working Conference on Graphic Languages* (North-Holland, Vancouver, Canada, 1972).

[15] Alexandre BRANDWAJN, *Équivalence et décomposition dans les modèles à files d'attente, et leur application à l'évaluation des performances des systèmes d'exploitation*, Thèse de doctorat ès sciences, Université Pierre et Marie Curie, Paris-6 (1975).

[16] Barbara S. BRAWN & Frances G. GUSTAVSON, « Program Behavior in a Paging Environment », *in Proceedings of the AFIPS Fall Joint Computer Conference*, p. 1019–1032 (1968).

[17] Keith G. CALKINS, « The Computer That Will Not Die : The SDS Sigma 7 », (1984), http ://www.andrews.edu/~calkins/profess/SDSigma7.htm.

[18] R. CHEVANCE, « Le projet Y », *in Quatrième Colloque sur l'histoire de l'informatique* (Rennes, 1995), http ://www.feb-patrimoine.com/histoire/english/projet_y.pdf.

[19] Fernando J. CORBATÓ, Marjorie MERWIN-DAGGETT & Robert C. DALEY, « An Experimental Time-Sharing System », *in Proceedings of the AFIPS Fall Joint Computer Conference, Part 1*, p. 335–344 (1962), reprinted in *Programming Systems and Languages* (S. Rosen, ed.), pages 683–698, McGraw-Hill, 1967.

[20] Fernando J. CORBATÓ & Victor A. VYSSOTSKY, « Introduction and Overview of the Multics System », *in Proceedings of the AFIPS Fall Joint Computer Conference, Part 1*, vol. 27, p. 185–196 (1965).

[21] CORNAFION (NOM COLLECTIF), *Systèmes informatiques répartis – concepts et techniques* (Dunod, 1981).

[22] CROCUS (NOM COLLECTIF), « Crocus, une étape dans l'enseignement des systèmes d'exploitation », *in Troisième Colloque sur l'histoire de l'informatique* (Sophia Antipolis, 1993),
http ://cnum.cnam.fr/RUB/histcrocus.pdf.

[23] Peter J. DENNING, « Thrashing : Its Causes and Prevention », *in Proceedings of the AFIPS Fall Joint Computer Conference*, p. 915–922 (1968).

[24] Jack B. DENNIS, « Segmentation and the Design of Multiprogrammed Computer Systems », *Journal of the ACM*, vol. 12 (4), (1965) p. 589–602.

[25] E. W. DIJKSTRA, « Cooperating sequential processes », *in* F. GENUYS, editor, *Programming Languages : NATO Advanced Study Institute*, p. 43–112 (Academic Press, 1968), [Initialement publié comme EWD-123, Lecture notes, Eindhoven, 1965]
http ://www.cs.utexas.edu/users/EWD/ewd01xx/EWD123.PDF.

[26] E. W. DIJKSTRA, « The Structure of the "THE" Multiprogramming System », *Communications of the ACM*, vol. 11 (5), (1968) p. 341–346.

[27] Jean FERRIÉ, *La gestion des processus dans un système à partage de ressources*, Thèse de docteur-ingénieur, Université Paul Sabatier, Toulouse (1971).

[28] P. FREEMAN, *Software Systems Principles : A Survey* (Science Research Associates, 1975), cité pp. 541, 631.

[29] E. GELENBE & C. KAISER, , *Operating Systems : Proceedings of an International Symposium*, vol. 16 de *LNCS* (Springer Verlag, IRIA, Rocquencourt, 1974).

[30] C. GIRAULT, « Un des systèmes de multiprogrammation réalisés à l'Institut de Programmation de la faculté des sciences de Paris », *RIRO*, vol. B-2, (1971) p. 3–18.

[31] S. GUIBOUD-RIBAUD & J. BRIAT, « Espace d'adressage et espace d'exécution du système GEMAU », *in* Gelenbe & Kaiser [29], p. 131–167.

[32] A. N. HABERMANN, *Introduction to Operating System Design* (Science Research Associates, 1976), 372 pp., cité pp. 84–85.

[33] C. KAISER & S. KRAKOWIAK, « Analyse de quelques pannes d'un système d'exploitation », *in* Gelenbe & Kaiser [29], p. 188–207.

[34] C. KAISER & S. KRAKOWIAK, « Design and Implementation of a Time-sharing System : A Critical Appraisal », *in* J. L. ROSENFELD, , *Information Processing, Proceedings of IFIP Congress 71*, p. 247–251 (North-Holland, Stockholm, 1974).

[35] Claude KAISER, *Conception et réalisation de systèmes à accès multiple : gestion du parallélisme*, Thèse de doctorat ès sciences, Université Pierre et Marie Curie, Paris-6 (1973).

[36] Sacha KRAKOWIAK, *Conception et réalisation de systèmes à accès multiple : allocation de ressources*, Thèse de doctorat ès sciences, Université Pierre et Marie Curie, Paris-6 (1973).

[37] C. J. KUEHNER & B. RANDELL, « Demand Paging in Perspective », *in Proceedings of the AFIPS Fall Joint Computer Conference*, p. 1011–1017 (1968).

[38] B. W. LAMPSON, « A Scheduling Philosophy for Multi-processing Systems », *Communications of the ACM*, vol. 11 (5), (1968) p. 347–359.

[39] B. W. LAMPSON, W. W. LICHTENBERGER & M. W. PIRTLE, « A User Machine in a Time-sharing System », *Proceedings of the IEEE*, vol. 54 (12), (1966) p. 1766–1774, reprinted in *Computer Structures*, ed. Bell and Newell, McGraw-Hill, 1971.

[40] A. LEMAIRE & F. PRUSKER, « Principes de fonctionnement et langage de spécification des interactions du système graphique MÉTAVISU », *in Actes du Congrès AFCET* (Grenoble, 1972).

[41] H. M. LEVY & P. H. LIPMAN, « Virtual Memory Management in the VAX/VMS Operating System », *IEEE Computer*, vol. 15 (3), (1982) p. 35–41.

[42] Stuart E. MADNICK & John J. DONOVAN, *Operating Systems* (McGraw-Hill, 1974), 640 pp., cité p. 574.

[43] Jacques MOSSIÈRE, *Allocation de ressources dans les systèmes à accès multiples*, Thèse de 3e cycle, Université Pierre et Marie Curie, Paris-6 (1971).

[44] Dennis W. RITCHIE & Ken THOMPSON, « The UNIX Time-Sharing System », *Communications of the ACM*, vol. 17 (7), (1974) p. 365–375, [Version étendue d'une communication présentée au *Fourth ACM Symposium on Operating Systems Principles*, Yorktown Heights (NY, USA), oct. 1973].

[45] Jean-Paul ROSSIENSKY & Vincent TIXIER, « A Kernel Approach to System Programming : SAM », *in* J. T. TOU, , *Software Engineering*, vol. 1, p. 205–224 (Academic Press, New York, 1969).

[46] Jean-Paul ROSSIENSKY & Vincent TIXIER, « LP70, a System Programming Language with Parallel Processes », *in Proc. International Computing Symposium*, p. 492–513 (Bonn, Germany, 1970).

[47] M. ROZIER, V. ABROSSIMOV, F. ARMAND, I. BOULE, M. GIEN, M. GUILLEMONT, F. HERRMANN, C. KAISER, P. LÉONARD, S. LANGLOIS & W. NEUHAUSER, « The Chorus Distributed Operating System », *Computing Systems*, vol. 1 (4), (1988) p. 305–379.

[48] J. H. SALTZER, *Traffic Control in a Multiplexed Computer System*, Ph.D. thesis, MIT (1966), tech. Report MAC-TR-30.

[49] Alan C. SHAW, *The Logical Design of Operating Systems* (Prentice Hall, 1974), 304 pp., cité p. 80.

[50] N. H. SHELNESS, P. D. STEPHENS & H. WHITFIELD, « The Edinburgh Multi-Access System Scheduling and Allocation Procedures in the Resident Supervisor », *in* Gelenbe & Kaiser [29], p. 293–310.

[51] A. J. SHILS, « The Load Leveler », rapport technique RC 2233, IBM T. J. Watson Research Center (1968).

[52] Laurent TRILLING & Jean-Pierre VERJUS, « An Attempted Definition of an Extensible System », *ALGOL 68 Implementation*, p. 119–139.

[53] Gerald M. WEINBERG, *The Psychology of Computer Programming* (Van Nostrand Reinhold, 1971).

[54] H. WHITFIELD & A. S. WIGHT, « EMAS : Edinburgh Multi-Access System », *The Computer Journal*, vol. 4, (1974) p. 331–346.

[55] Niklaus WIRTH, « PL360, a Programming Language for the 360 Computers », *Journal of the ACM*, vol. 15 (1), (1968) p. 37–74.