Qu'est-ce qu'un système d'exploitation ? Une approche historique

Maarten Bullynck

Qu'est-ce qu'un système d'exploitation ? Une approche historique

Maarten Bullynck
Centre de recherches historiques : Histoire des pouvoirs, savoirs et sociétés (EA 1571), Université Paris 8.

Résumé

Les débuts des systèmes d'exploitation sont souvent obscurcis par leur proximité aux systèmes de programmation et ont été minimisés dans la suite des discussions des années 1960 entre les défenseurs du traitement par lots et les partisans du partage de temps. Une étude plus détaillée et plus systématique des systèmes de contrôle entre 1954-1964 montre pourtant qu'il existe une variété d'idées, de philosophies et d'implémentations qui ont souvent préparé la voie à des systèmes ultérieurs. Cet article offre d'abord une synthèse de développements pertinents en matériel et en logiciel et propose ensuite une taxonomie de systèmes de contrôle pendant la période 1954-1964. Enfin, les origines du terme operating system et sa dissémination sont retracées dans la communauté des utilisateurs des machines d'IBM.

Mots-clés : systèmes d'exploitation, histoire de l'informatique, organisation du travail informatique, partage de temps, programmation automatique.

Un système d'exploitation est aujourd'hui devenu indispensable : on ne peut guère s'imaginer un ordinateur sans un système d'exploitation qui gère et façonne l'accès à l'ordinateur et à ses périphériques, et sans que l'interaction entre l'ordinateur et ses utilisateurs ne passe par cette interface parfois devenue invisible[1]. Pourtant, quand les premiers ordinateurs sont apparus après la seconde Guerre Mondiale, il n'y avait rien de tel. Il a fallu une décennie avant que les premiers essais de système d'exploitation ne soient formulés et implémentés. Il a fallu une décennie supplémentaire pour que l'idée soit généralement acceptée et que presque tous les ordinateurs soient vendus ou loués en incluant un système d'exploitation. Il faut attendre la fin des

1 Cet article est une adaptation libre d'un article qui apparaîtra dans un volume spécial de HAPOP-3.

années 1960, avec le développement des systèmes d'exploitation OS/360 par IBM et de Multics par une équipe associant le M.I.T., les Bell Labs et General Electric, pour que des principes généraux et des cadres théoriques pour les systèmes d'exploitation soient mis en place. L'émergence des systèmes de partage de temps (*time-sharing*) est considérée traditionnellement comme un tournant dans l'histoire des systèmes d'exploitation. Les discussions parfois acérées entre les partisans du *time-sharing* et les défenseurs du traitement par lots (*batch processing*) dans les années 1965-1975[2] sont devenues légendaires et font partie intégrante de l'histoire de l'informatique. Mais ces discussions, aussi importantes soient-elles pour repenser la relation entre ordinateur et utilisateur, et révélatrices aussi d'un « *combat gargantuesque pour dominer le marché du logiciel qui se chiffre en milliards* »[3], ont biaisé l'historiographie des systèmes d'exploitation.

L'histoire classique de ces systèmes part de l'ordinateur sans système d'exploitation, passe par le traitement par lots pour arriver enfin à la multiprogrammation ou le *time-sharing* moderne[4]. C'est une histoire où une configuration variable de programmeurs, opérateurs et ingénieurs travaillant avec un ordinateur est rationalisée à l'aide du traitement par lots. Une nouvelle configuration, qui automatise partiellement le programmeur et l'opérateur, se met en place : le travail des programmeurs est séparé du travail opératoire. Les programmeurs, travaillent dans leurs bureaux, écrivent leurs programmes puis les apportent à l'ordinateur et ses opérateurs. Les programmes sont mis en lots et attendent leur exécution, qui est gérée par l'opérateur et aidée par un système d'exploitation. Cette histoire tourne aussi, sans l'évoquer explicitement, autour de l'entreprise IBM. Le traitement par lots a débuté sur des machines d'IBM et quand commencent les débats entre partisans du *batch-processing* et du *time-sharing* au début des années 1960, IBM sera lent à repérer le *time-sharing* et on lui reprochera de rester plutôt fidèle au traitement par lots. Ce sont tous des fils pertinents de l'histoire des systèmes d'exploitation, mais ils cachent la variété et la complexité des systèmes issus de besoins et visions très différents des utilisateurs de la période allant de 1954 à 1964. Cette période n'est pas seulement l'ère des systèmes de traitement par lots. Au contraire, il existait plusieurs modes possibles d'utilisation d'un ordinateur et plusieurs concepts pour automatiser en partie cette utilisation. En outre, l'idée même d'un système d'exploitation n'était pas stabilisée.

Dans ce qui suit on explore des questions et problèmes qui ont stimulé

2 Un échantillon caractéristique de cette discussion se trouve dans le numéro spécial de Septembre 1965 du journal *Computers and Automation*.

3 « […] *a gargantuan contest to dominate the multibillion dollar software industry* » (Sackman, 1970, p. 8).

4 Ce récit classique se trouve chez (Ceruzzi, 2003, pp. 96-101), (Tanenbaum, 2001, pp. 6-18) ou (Krakowiak, 2013, 2014). Le livre de Brinch-Hansen (2001) suit aussi cette chronologie, mais sa présentation met en évidence que les systèmes et leurs « philosophies » se développent en parallèle.

les débuts des systèmes d'exploitation[5]. Cette démarche permet d'explorer des couches historiographiques et épistémologiques qui approfondissent le récit classique tout en mettant en valeur des développements en parallèle et en restant focalisé sur cette question : qu'est-ce qu'un système d'exploitation ?

Prérequis technologiques pour les systèmes d'exploitation (1954-1964)

Débuts dans les années 1950

L'idée même que la machine est en charge de sa propre programmation est en fait implicite dans le modèle de l'ordinateur à programme enregistré (*stored-program computer*). Parce que l'ordinateur calcule beaucoup plus vite que l'homme, l'ordinateur doit contrôler et gérer l'exécution des programmes. L'étape suivante conduirait logiquement à l'idée qu'un programme peut contrôler et gérer les autres programmes. Pourtant, dans les années 1946-1956 peu d'ordinateurs étaient capables d'implémenter cette idée : tout au plus, des routines préparatoires ou des routines d'amorçage (*bootstrapping routines*) étaient disponibles. Au milieu des années 1950, cette situation change pour de nombreuses raisons. La diversification de l'utilisation de l'ordinateur, en particulier son usage croissant pour le traitement des données (*data processing*), combiné avec des développements technologiques et des développements logiciels font naître une variété de systèmes pour exploiter un ordinateur.

Côté technologie, les tambours magnétiques (*magnetic drum memory*) deviennent populaires en tant que mémoire de travail, et parfois comme mémoire de stockage. Les tores magnétiques (*magnetic ferrite core memories*) sont développés au M.I.T. au début des années 1950, technologie qui rend possible une mémoire de travail plus stable, plus rapidement accessible. Les tambours magnétiques sont surtout populaires dans les années 1950, les tores magnétiques par contre, étant plus chers à leur début, ne se répandront vraiment que dans les années 1960. Les bandes magnétiques constituent la dernière technologie importante. Cette technologie, proposée dès 1951, permet une mémoire de stockage plus rapide et plus importante. Lentement, les bandes magnétiques vont remplacer les cartes perforées et les bandes de papier perforées, même si jusque dans les années 1960 ces médias de stockage sont encore utilisés en parallèle des bandes magnétiques. Les tambours magnétiques et les bandes magnétiques permettent un accès plus rapide aux données enregistrées et facilitent donc l'accès à un grand nombre de routines (une bibliothèque de routines). Comparé aux cartes perforées, cet accès n'est pas seulement plus rapide mais aussi plus automatisable et donc moins assujetti à

5 Cette analyse s'appuie sur une étude systématique des premiers systèmes qui sera publiée ailleurs.

l'intervention humaine. Évidemment, le mode d'accès aux données est soumis aux propriétés physiques des supports : la lecture et l'écriture se font de manière séquentielle (pour les bandes) ou cyclique (pour les tambours). Un accès direct (*random access*) deviendra possible seulement dans les années 1960, avec les tores magnétiques et les disques durs (*disk drive*).

L'introduction des mémoires tampons (*buffer memory*) pour la communication entre le processeur central et les organes d'entrée et de sortie constitue une autre avancée technologique. Avant ces mémoires, le processeur devait attendre la fin d'une lecture ou d'une écriture, ou d'une communication avec un dispositif d'entrée ou de sortie, pour passer à l'étape suivante. Cette forme de communication limitait la vitesse du processeur et pouvait même la réduire à la vitesse de fonctionnement des organes d'entrée ou de sortie. Si on sait par exemple que sur l'ENIAC le processeur pouvait opérer 5 000 additions par seconde, le lecteur de cartes perforées ne pouvait lire que 2 cartes par seconde : il est facile d'imaginer que cette inadéquation entre vitesses pouvait ralentir sérieusement l'opération d'un ordinateur. De nombreuses stratégies avaient été élaborées pour éviter ce blocage, utilisant des bandes magnétiques comme mémoire tampon, ou même utilisant tout un ordinateur pour traiter les communications avec des périphériques entrée et sortie. Mais avec l'introduction systématique des mémoires tampon pour entrée et sortie, « *l'entrée et la sortie ont quitté le domaine des vitesses mécaniques pour accéder au domaine des vitesses électroniques* »[6].

L'expansion du stockage rapide pour les programmes va de pair avec le développement du logiciel (*software*). Les routines programmées sont cumulativement rassemblées dans une bibliothèque, stockées dans une mémoire externe. Entre utilisateurs d'un même type d'ordinateur s'installent des procédures et des usages de partage et de recyclage. La fondation des groupes d'utilisateurs comme SHARE pour les utilisateurs du IBM 701 ou de USE pour les utilisateurs du UNIVAC 1103 (tous les deux en 1955) marque le moment où le logiciel devient un article de valeur. Des groupes d'utilisateurs se réunissent régulièrement pour partager des programmes et pour discuter des pratiques de programmation. Si les programmes dans ces groupes étaient partagés gratuitement, la deuxième moitié des années 1950 voient également les débuts d'une industrie du logiciel (Cerruzzi, 2003, pp. 79-108). Des sociétés comme System Development Corporation (SDC, 1957) ou Computer Sciences Corporation (CSC, 1959) louaient leurs services aux grands projets du gouvernement ou aux grandes entreprises (Campbell-Kelly, 2001, pp. 29-56).

6 « *With the advent of this phase [I/O buffer memory], input-output was taken out of the domain of mechanical speeds and placed in the domain of electronic speeds* » (Bauer, 1958).

Dans ce contexte, les premiers langages de programmation sont développés, des bibliothèques de routines s'accumulent, et on voit aussi l'apparition de systèmes de programmation qu'on peut appeler, *a posteriori*, des systèmes d'exploitation. L'un des premiers systèmes, qui sera très influent, a été le *Comprehensive System of Service Routines* (CSSR) développé au Lincoln Lab du M.I.T. pour leur ordinateur Whirlwind (1953). Ce système sera plus tard le fond sur lequel SDC va ériger son système de programmation pour le projet SAGE. Un autre type de systèmes importants, les moniteurs de séquences de programmes (*monitor program*) faits pour enchaîner automatiquement l'exécution de programmes, sont développés au cœur de la communauté de SHARE. Ces moniteurs deviendront plus tard les noyaux pour les systèmes de traitement par lots (*batch processing*), très typiques pour les installations commerciales et scientifiques des machines IBM dans les années 1960.

Deux innovations ponctuelles, l'une en technologie, l'autre en logiciel, sont cruciales pour l'évolution des systèmes d'exploitation. D'abord, en 1956, l'interruption (*interrupt*) était introduite sur le ERA 1103A sur la demande d'un utilisateur, la NSA[7]. L'interruption pouvait interrompre la marche de la machine pour faire communiquer le processeur central avec un périphérique. En utilisant l'interruption, les interruptions manuelles effectuées par l'opérateur humain pouvaient être (partiellement) automatisées. Par cette voie, des moniteurs plus compliqués étaient envisageables et la multiprogrammation, et plus tard le partage de temps, devenait possible. Puis, en 1957 est sorti FORTRAN (pour FORmula TRANslation), aboutissement du travail d'un groupe de programmeurs pour IBM visant à créer un langage scientifique de programmation. En tant que premier langage de programmation complet[8], FORTRAN (puis FORTRAN II) devint vite populaire et presque indispensable pour beaucoup d'installations d'ordinateur. Pour beaucoup de systèmes de programmation existants, l'apparition de FORTRAN a mené à des réécritures considérables afin de pouvoir l'intégrer. Un bon nombre de systèmes d'exploitation autour de 1960 sont (re)développés par de grands utilisateurs institutionnels pour inclure FORTRAN, par exemple le FORTRAN Monitor System (FMS, 1959) développé par North American Aviation, ou BESYS-3 (1960) par Bell Labs, UMES (1959) par l'université de Michigan ou encore le RAND-SHARE Operating System (1962) développé par la RAND Corporation.

7 Cet ordinateur est parfois aussi connu sous le nom de Scientific Univac 1103, et il fut d'abord développé pour la NSA sous le nom Atlas II.

8 Il y a d'autres langages de programmation développés avant ou en parallèle de FORTRAN, voir (Knuth & Pardo, 1977), mais ou bien leur utilisation fut limitée, ou bien il leur manquait souvent des fonctionnalités importantes, comme par exemple des commandes pour programmer les périphériques.

Changements au milieu des années 1960

Les années entre 1962 et 1964 marquent la fin d'une première phase dans le développement des systèmes d'exploitation. Les « grands » projets de systèmes d'exploitation comme OS/360 d'IBM ou Multics, et en particulier l'émergence des systèmes à temps partagé, symbolisent ce tournant, même s'ils sont plutôt les traits les plus visibles d'une évolution plus générale. Cette évolution est portée, d'une part, par le développement graduel de la multiprogrammation, et d'autre part, par l'introduction de nouveaux types de mémoire de stockage. La multiprogrammation, en essence, repose sur l'idée que plusieurs programmes peuvent être exécutés en même temps sur une machine, ce qui rompt avec le traitement séquentiel inhérent à l'architecture de von Neumann, où une instruction est exécutée seulement après que la précédente soit terminée. En pratique, la multiprogrammation n'est souvent qu'une répartition du temps du processeur central sur plusieurs programmes. Le processeur de l'ordinateur traite encore séquentiellement les programmes, mais, en utilisant des mémoires tampons (*buffer memories*), des programmes attendent leur tour, ou sont interrompus pour être repris plus tard. L'ordonnancement (*scheduling*) organise l'ordre d'exécution de (morceaux) de programmes. L'interrupteur (*hardware interrupt*) a rendu possible les premières tentatives de multiprogrammation, et l'introduction des mémoires tampon pour entrée et sortie a fait proliférer la multiprogrammation sous des formes variées, dont l'une des plus extrêmes est le partage de temps (*time-sharing*). Pourtant, la transition d'une mémoire de stockage séquentielle à une mémoire vive à accès direct (*random access*) a peut-être eu l'influence la plus profonde sur l'implémentation des systèmes d'exploitation dans les années 1960. Cette transition a permis d'abandonner la logique séquentielle des bandes magnétiques et d'initier l'accès direct. Le disque dur du RAMAC d'IBM (1956) est le premier exemple d'une mémoire vive. Les disques IBM 1405 et IBM 1301 (1961-1962), développés pour être utilisés sur l'IBM 1410 et la ligne IBM 7000, ont été les systèmes les plus répandus.

Mais il n'y a pas seulement des avancées technologiques. Entre 1962 et 1964 il semble que presque tous les producteurs d'ordinateurs ont repris l'idée d'un système d'exploitation et l'ont développé pour l'inclure dans le paquet ordinateur-logiciel qui était loué ou vendu[9]. Avant 1960, le développement des systèmes d'exploitation était assuré en majorité par les utilisateurs des systèmes d'ordinateurs et dans quelques contextes bien particuliers (surtout des projets de recherche financés par le militaire). Après 1960, ce sont les entreprises qui prennent en main le logiciel (*software*), elles embauchent des programmeurs et investissent dans

9 C'est également au début des années 1960 que les premiers articles de synthèse sur les systèmes d'exploitation apparaissent, notamment (Orchard-Hays, 1961, pp.290-294 ; Mealy, 1962).

la conception d'outils de programmation pour leurs machines. Ces outils incluent des bibliothèques de routines, des (macro) assembleurs, des compilateurs pour des langages de programmation (comme FORTRAN ou COBOL), des langages de programmation, des outils de *debugging*, mais aussi des systèmes pour organiser des fichiers ou des données, des « routines de maître » (*master routines*) et des systèmes d'exploitation. Si on regarde quelques-unes des principales sociétés de systèmes informatiques, elles sortent toutes leur propre système d'exploitation entre 1962 et 1965 (voir Table 1). Certains de ces systèmes sont plutôt rudimentaires (comme BRIDGE de General Electric), d'autres sont des systèmes de traitement par lots classiques (comme BKS de Philco ou Scope de CDC), mais la majorité ont des options avancées de multiprogrammation combinées avec le traitement par lots. Le partage de temps, par contre, est encore en développement et existe avant 1966 surtout dans des contextes de recherche, par exemple CTSS sur un IBM 709 et plus tard sur un PDP-1 (M.I.T.), TS sur un PDP-1 (BBN), JOSS sur le Johnniac (RAND), le Dartmouth Time-Sharing System (Dartmouth College), le Cambridge multiple-access System sur le Ferranti Atlas (Cambridge), etc. Dans des installations commerciales, le partage de temps fait son apparition vers la fin des années 1960 quand IBM, GE, DEC, PDP, SDS et d'autres sociétés vont l'incorporer dans leurs systèmes d'exploitation.

Société	Ordinateur	Année	Système d'exploitation
Honeywell	H800	1961	Executive Monitor
Univac	Univac 1107	1962	EXEC I
Burroughs	D825	1962	AOSP
Burroughs	B5000	1962	Master Control Program
Philco	Philco-2000	1962	SYS; BKS
GE	GE-215/225/235	1962	BRIDGE
IBM	IBM 7090/7094	1962	IBSYS
CDC	CDC 1604	1962	CO-OP monitor system
CDC	CDC 3600	1963	SCOPE monitor system
Honeywell	Honeywell 1800	1963	ADMIRAL master monitor
GE	GE 625-635	1964	Comprehensive Operating Supervisor
RCA	RCA 3301	1964	Realcom System
DEC	PDP-6	1964	Supervisory Control Program
SDS	SDS 9000	1964	MONARCH

L'évolution d'un développement par l'utilisateur vers des systèmes faits maison par les sociétés elles-mêmes est la plus apparente chez IBM. Les premiers systèmes d'exploitation pour des machines IBM sont développés par des utilisateurs comme General Motors, North American Aviation, Bell Labs, Michigan University, etc. Ils s'appuyaient sur le partage d'idées et de programmes au sein de la communauté SHARE d'utilisateurs de machines IBM, mais ne recevaient pas de support direct d'IBM même. Lentement, IBM en tant que société s'est impliquée également. Le constructeur a aidé le développement de Share Operating System (1959) qui est né au cœur de SHARE. Plus tard, il va incorporer le FORTAN Monitor System, initialement développé par North American Aviation, dans son système de programmation FORTRAN pour le IBM 709/7090 (1960). Finalement, IBM commence à concevoir ses propres systèmes d'exploitation, d'abord IBSYS (à partir de 1962), puis OS/360 (à partir de 1965). En parallèle, le développement par les utilisateurs disparaît lentement, même si certains utilisateurs vont encore bricoler avec les systèmes des fabricants et les adapter à leurs besoins[10].

Qu'est-ce qu'un système d'exploitation ? Des philosophies différentes

Si on utilise le terme usuel en anglais pour système d'exploitation, *operating system*[11], on souscrit implicitement à l'idée qu'un système d'exploitation automatise la mise en opération de l'ordinateur et remplace, en partie, le travail d'un opérateur ou d'une opératrice humaine. On peut l'appeler la philosophie opérationnelle du système d'exploitation. En particulier, les opérations à exécuter manuellement sur un panneau de contrôle (*control panel*), ou panneau moniteur (*monitor panel*) ou panneau de supervision (*supervisory panel*) sont automatisées par l'*operating system*. Sur ces panneaux, l'opérateur pouvait agir quand il y avait des arrêts (après l'exécution d'un programme ou après la fin d'une lecture ou d'une écriture par un périphérique, ou quand il faut attendre que l'opérateur prenne la bande magnétique de la bibliothèque de routines dont on a besoin) ou des interruptions (quand un programme ou un périphérique ne fonctionne pas comme prévu, ou quand l'instruction ne peut pas être exécutée), ou d'autres signaux encore. L'*operating system* automatise les réponses de l'opérateur à ces arrêts, interruptions et signaux. Par cette auto-

10 Un exemple est Thompson-Ramo-Woolridge qui va développer un système d'exploitation à partir de IBSYS en 1962 (Nelson, 1969).

11 La plupart des langues modernes utilisent une variante de *operating system*, mais pas tous, par exemple le français (système d'exploitation), l'allemand (*Betriebssystem*), le néerlandais (*besturingssysteem*) ont d'autres termes.

matisation, il devient possible que toute une séquence, un lot (*batch*) de programmes, puisse être accomplie sans interruption entre deux arrêts, et non plus seulement un programme à la fois : d'où le nom « traitement par lots » (*batch processing*) pour cette première génération de systèmes d'exploitation. Ce type de système pouvait réduire le temps inutilisé de l'ordinateur et accélérer le chargement des programmes. En outre, certaines erreurs humaines pouvaient être évitées et on pouvait automatiser les processus de chargement et de traduction, une fois qu'ils étaient standardisés et codés dans un format fixe. Dans ce contexte, on dit souvent à l'époque que le système d'exploitation fait le « ménage » (*housekeeping*).

L'automatisation partielle de l'opérateur humain a mené aussi à une autre configuration et opération dans les centres de calcul. Traditionnellement, on parle de la transition d'*open shop* (opération ouverte du centre de calcul) à *closed shop* (opération fermée)[12]. Dans la configuration *open shop*, on emmenait son programme au centre de calcul, laissait l'opérateur mettre le programme sur l'ordinateur, et après exécution du programme, on pouvait prendre les résultats et retourner dans son bureau. La configuration *closed shop* par contre réduisait le contact avec l'ordinateur et son opération, l'opérateur humain recueillait les programmes et les mettait en lots. On devait attendre l'exécution du lot dans lequel était son programme pour pouvoir chercher les résultats dans le centre de calcul. La configuration *closed shop* est donc intimement liée à la philosophie de traitement par lots[13].

On cite souvent 1956 comme l'année de naissance pour les systèmes d'exploitation du type traitement par lots, mais la généalogie commence un peu plus tôt avec des systèmes comme le 701 Monitor (1955) de Owen Mock à North American Aviation (Mock, 1987) ou le Supervisor pour l'IBM 702 de Bruse Moncreiff à RAND (Moncreiff, 1956). L'idée mûrit avec le développement en 1956 du General Motors-North American Aviation Monitor (sous l'acronyme GM/NAA Monitor) pour un IBM 704, moniteur qui sera partagé dans la communauté SHARE (Patrick, 1987). Le noyau de ce système, le Mock-Donald monitor, sera recyclé, adapté, actualisé et implanté dans des systèmes d'exploitation plus complexes et plus ambitieux comme le SHARE Operating System

12 Les termes *open shop* et *closed shop* sont empruntés aux pratiques des syndicats aux États-Unis. En *closed shop*, les ouvriers devraient être syndiqués pour faire une tâche et il y avait des restrictions sur le type de tâche qu'un ouvrier pouvait faire.

13 Parfois, une autre interprétation est donnée à *open shop* versus *closed shop*. Dans cette interprétation, le *closed shop* est la situation où seulement les opérateurs et les programmeurs en code machine peuvent utiliser la machine, parce que les autres utilisateurs ne savent pas écrire des programmes en code machine ou assembleur. L'*open shop* est alors le cas où les utilisateurs peuvent écrire leurs propres programmes, dans un langage de programmation. Ces programmes peuvent être mis dans des lots, voir (Breheim, 1961) pour un exemple d'*open shop* sous le système d'exploitation FMS qui organise les tâches en lots.

pour l'IBM 709 (SOS, 1959) ou le RAND-SHARE Operating System pour l'IBM 7090 (1962). Dans le système d'Owen Mock, des programmes étaient mis dans un lot dont la durée d'exécution était estimée à une heure à peu près. Dans le GM/NAA Monitor, le traitement devenait plus complexe et se faisait en trois phases. D'abord une traduction pour convertir les données décimales en binaire et les programmes en instructions machine ; puis l'exécution du programme sous le contrôle du programmeur ; et finalement la sortie des résultats, imprimés ou perforés, après traduction[14]. L'évolution vers des moniteurs de plus en plus complexes sera poursuivie et trouvera son apogée en IBSYS (1962-1965) qui comme « *moniteur de moniteurs* […] *contient plusieurs des anciens systèmes* » (Hassitt, 1967, p. 24).

Toutefois, même si dans les histoires classiques des systèmes d'exploitation le type « traitement par lots » et la philosophie opérationnelle dominent le récit avant l'arrivée des systèmes à temps partagé, ce ne sont pas les seuls systèmes qui existent avant 1964. Une autre philosophie, très influente, est par exemple portée par le Comprehensive System of Service Routines (CSSR) du M.I.T. pour leur ordinateur Whirlwind (Bennington & Gaudette, 1956). Ce système, en développement continu depuis 1953, avait l'ambition de faciliter de manière générale l'accès à plusieurs groupes de programmes et d'automatiser certaines tâches du programmeur humain. Il y avait des groupes de programmes de lecture et écriture, de traduction et conversion, de diagnostic et de *debugging*, de contrôle et de moniteur, etc. L'idée derrière le *comprehensive system* est généraliste et inclut à la fois le(s) système(s) de programmation, le système d'exploitation et la bibliothèque de routines. Cette philosophie correspond à une approche plus unitaire et interconnectée d'organiser les logiciels autour de la machine.

On peut appeler cette philosophie « intégrative », d'après le terme *integrated system* souvent utilisé dans les années 1950 et 1960. Le terme est particulièrement populaire chez les personnes qui viennent du projet Whirlwind ou de Ramo-Woolridge. Dans le *Handbook for Automation, Computation and Control* (1959), le système intégré est défini comme « *l'interconnexion de quelques ou tous les programmes de service dans une unité organisée, contrôlée par le programmeur et semi-automatique ou complètement automatisée* »[15], ou encore : « *tous les programmes sont inté-*

14 « *an input-translation phase which converted data from decimal to binary, and programs from source to object language ; an execution phase which was almost exclusively under the programmer's direct control ; and an output translation phase which processed line printer output, punched card output (both decimal and binary), and accounting records* » (Patrick, 1987, p. 802).

15 « *Interconnection of some or all these different utility programs into an organized, programmer-controlled, semiautomatic or automatic whole is usually called an integrated system* » (Grabbe & *al.*, 1959, p. 184).

grés pour faire un seul système de calcul afin d'exclure l'utilisation de la machine par des sous-systèmes isolés »[16]. Des exemples de ce genre de système sont le CSSR du M.I.T., *MAGIC (Michigan Automatic General Integrated Computation)* de l'université de Michigan ou le système intégré développé à Ramo-Woolridge par le groupe de W.F. Bauer pour l'ordinateur ERA 1103 (1955).

Dans la philosophie des systèmes intégrés, tout est organisé autour de l'interconnexion entre programmes, ce qui contraste avec la philosophie opérationnelle qui sépare les logiciels aidant le programmeur des logiciels utilisés par l'opérateur. Aussi le(s) programme(s) qui contrôle(nt) la séquence des opérations n'ont pas nécessairement un statut à part. La bibliothèque des routines, en particulier les routines de service (*utility programs*), est le noyau du système. Le système d'exploitation du type intégré est surtout conçu comme structure d'accès à la machine par des groupes de routines qu'on peut combiner, varier et développer. C'est un aspect qui est également présent dans les systèmes d'exploitation modernes[17]. À cause de cette vision, ce type de systèmes avait tendance à ne pas toujours utiliser le cycle linéaire « *charger, assembler, compiler et exécuter* » (load, assemble, compile and execute) et à s'appuyer sur des routines interprétatives. Contrairement aux langages de programmation dans lesquels tout un programme est d'abord compilé pour être exécuté, dans un système interprétatif une ligne d'un programme est interprétée et directement exécutée. Ce système rend possible le remplacement du code de la machine par un langage plus avancé de sous-routines, et au lieu d'avoir plusieurs « *instructions de saut à une sous-routine* », on a « *une seule sous-routine qui interprète et supervise toutes les sous-routines appelées* »[18]. Sur beaucoup d'ordinateurs des années 1950, on disposait de systèmes interprétatifs pour faire des calculs en virgule flottante. Sur quelques ordinateurs avec beaucoup de mémoire de travail et des outils d'entrée et de sortie comme un *flexowriter* ou un écran cathodique, les systèmes interprétatifs préfiguraient parfois les systèmes interactifs qui naîtront avec le temps partagé. C'est le cas notamment pour le TX-0 ou le TX-2 développé par le Lincoln Lab au M.I.T.

16 « *The important concept here is that all items are integrated together to form one computation system to the exclusion of the use of the machine with isolated subsystems* » (Bauer, 1956, p. 8).

17 Cf. la remarque de G.H. Mealy : « *Many functions now classed as OS functions were first embodied as utility subroutines and programs* […] *Today, the library is an integral part of the OS – to the extent, for instance, that many programmers identify the UNIX system with its library rather than with its nucleus and shells* » (Mealy, 1987, p. 781).

18 « *The jump instructions in the main program which formerly directed control to the subroutines are eliminated* […] [the subroutines] *are all welded into one, an interpretive subroutine, which includes also a section to supervise the sequence in which the various operations are performed* », autrement dit « *the instruction code of the machine is not merely augmented, it is entirely replaced* » (Adams, 1954, pp. 16-3).

Aujourd'hui, on a du mal à distinguer clairement entre cette vision d'un système intégré et les systèmes de programmation. En général, il est, avant 1964, souvent difficile de bien distinguer entre ce qui relève d'un système de programmation ou d'un système d'exploitation. En effet, du point de vue de la philosophie intégrative, les routines de supervision, de moniteur ou de contrôle font, elles aussi, simplement partie de la bibliothèque des routines disponibles. Dans l'autre point de vue, les routines de moniteur supervisent et contrôlent les systèmes de programmation, elles sont donc hiérarchiquement au-dessus du système de programmation.

Même si la plupart des systèmes entre 1954 et 1964 peuvent être rangés dans les catégories de la philosophie opérationnelle ou de la philosophie intégrative, il existait aussi, comme le rappelle W.F. Bauer, « *un nombre de systèmes spécialisés, en particulier des systèmes de commande et contrôle qui utilisaient des idées avancées en systèmes d'exploitation* »[19]. Le système le plus connu et le plus influent est le système SAGE (*Semi Automatic Ground Environment*). SAGE était un projet majeur financé par l'armée américaine pour développer un système d'ordinateurs connectés par des lignes téléphoniques qui devait coordonner les informations venant des radars et des sites militaires afin d'obtenir une vue d'ensemble des activités aériennes. Le système devrait rassembler en temps réel toutes les informations nécessaires pour prendre des décisions militaires en cas d'une attaque (possiblement atomique) de l'URSS. Le Lincoln Lab du M.I.T. faisait partie du projet avec ses ordinateurs Whirlwind et TX-2. IBM a développé des ordinateurs AN-FSQ7 pour le projet. Plusieurs programmeurs de la RAND Corporation ont fondé la première société informatique du logiciel, SDC (*System Development Corporation*), pour écrire les programmes nécessaires pour le projet SAGE. Le projet a donc stimulé des innovations technologiques et en particulier en logiciel. Les systèmes développés par le M.I.T. et IBM contenaient des nouveautés, comme par exemple un sous-système interactif entre l'utilisateur et une visualisation de données sur un écran cathodique ; ou encore un sous-système de *teleprocessing*, l'utilisation des lignes téléphoniques pour transférer des données d'un ordinateur (ou périphérique) à un autre. D'une certaine manière, on pourrait appeler ces systèmes des systèmes distribués parce qu'un (ou plusieurs) processeur(s) central(ux) est couplé avec une variété de périphériques et qu'une communication en temps réel entre toutes ces unités est établie. Sur ces systèmes, beaucoup d'idées ont été développées et testées qui seront plus tard précieuses pour le développement des systèmes de multiprogrammation. Mais, à l'époque, ces systèmes étaient définis d'abord par leur caractère temps réel (parfois aussi appelé « accès direct » ou « en ligne »).

19 « […] *a number of special purpose systems, particularly command and control systems that utilized advanced operation system ideas ahead of their time* » (Bauer, 1972, p. 999).

Il y a eu d'autres systèmes spécialisés sans rapport avec le projet SAGE. Outre des applications militaires spécialisées, il y a eu surtout le développement de systèmes pour gérer en temps réel des usines. On citera dans ce domaine Thompson-Ramo-Woolridge et son RW-400, ainsi que General Electric qui avait créé son système GARDE pour l'ordinateur GE-312 spécialement pour le contrôle des générateurs d'électricité (1959). IBM aussi, utilisant l'expérience du projet SAGE, proposait des systèmes commerciaux de *teleprocessing* en temps réel. Le plus connu est le système SABRE pour la réservation de billets d'avion, mais il y a eu également des systèmes pour les sociétés des chemins de fer (TOPS) et d'autres encore.

Quand en 1966 IBM présente son système d'exploitation OS/360, on appellera ce système « de deuxième génération ». La première génération était composée des systèmes de traitement par lots, la seconde de l'OS/360 qui combinait l'idée de traitement par lots avec le caractère temps réel[20]. En effet, OS/360 combinait les deux lignes de développement au sein d'IBM : d'une part le traitement par lots né dans la communauté SHARE, d'autre part les systèmes en temps réel et de *teleprocessing* qu'IBM avait d'abord développé pour le projet SAGE avant de le commercialiser.

> Avec l'extension de l'utilisation des ordinateurs en télécommunications, la fonction de programme de supervision s'est élargie pour servir de pont entre traitement par lots et service à distance. Le programme de supervision qui en a résulté peut être utilisé pour contrôler un système qui traite des programmes en lots, ou un système qui contrôle plusieurs dispositifs de télécommunications, ou toute combinaison des deux[21].

20 « *Then, as now, the operating system aimed at non-stop operation over a span of many jobs and provided a computer-accessible library of utility programs. A number of operating systems came into use during the last half of the decade. In that all were oriented toward overlapped setup in a sequentially executed job batch, they may be termed 'first generation' operating systems. A significant characteristic of batched-job operation has been that each job has, more or less, the entire machine to itself, save for the part of the system permanently resident in main storage. During the above-mentioned period of time, a number of large systems-typified by SAGE, MERCURY, and SABRE-were developed along other lines; these required total dedication of machine resources to the requirements of one 'real-time' application. By and large, however, these real-time systems bore little resemblance to the first generation of operating systems, either from the point of view of intended application or system structure. Because the basic structure of OS/360 is equally applicable to batched-job and real-time applications, it may be viewed as one of the first instances of a 'second-generation' operating system. The new objective of such a system is to accommodate an environment of diverse applications and operating modes* » (Mealy, 1966).

21 « […] *as the use of computers extended into telecommunications, the function of the supervisory control program broadened to serve as the bridge between batch processing and service to remote locations. The resulting supervisory control program can be used to control a system which processes batch programs only, or a system dedicated to the control of telecommunications devices, or any combination of these two* » (Dines, 1966).

L'invention de l'*operating system* chez IBM

Bien que le terme *operating system* soit aujourd'hui prévalent dans la majorité des langues, plusieurs termes étaient en usage dans les années 1950 et 1960. Orchard-Hays dans sa synthèse sur les systèmes de programmation et d'exploitation de 1961 remarque :

> De nombreux termes sont utilisés pour les parties d'un système d'exploitation. Le mot « programme superviseur » est déjà apparu [antérieurement dans le texte]. Le superviseur est un programme qui garde le contrôle de la machine à tout moment et auquel retourne le contrôle après qu'une routine se termine ou s'arrête de manière inattendue. Les mots utilisés plus ou moins comme synonymes de « superviseur » sont : « routine exécutive », « moniteur », « routine du contrôle maître[22].

Plusieurs variantes de ces noms existent, telles que « routine de contrôle des séquences », « contrôle exécutif », etc. Les noms de ces routines désignent souvent l'ensemble du système[23]. On parlait de « système exécutif », « système superviseur », « système de contrôle », « système à séquencer des programmes » etc., et rarement de « système d'exploitation » (*operating system*).

Alors comment *operating system* est-il devenu le terme usuel ? Le terme semble avoir ses origines dans la communauté SHARE des utilisateurs des machines IBM. La première utilisation remonte à 1959 dans le contexte de développement du SHARE Operating System (SOS) piloté par un comité de SHARE et aidé par IBM. Dans les publications officielles dans le journal de l'ACM, le terme *operating system* n'est pas utilisé pour décrire le système de SHARE, ils parlent plutôt de SHARE 709 System[24]. Comme le note D.L. Shell qui présidait le comité de développement, « *le problème initial pour le comité était la définition même de ce que veut dire 'système'* »[25]. La partie du système qui contrôle les autres programmes est appelée *supervisory control program*, qui « *coordonne les parties du système SHARE 709 et est responsable du fait que l'ordinateur travaille de manière ininterrompue pendant le traitement d'un groupe de tâches indépendantes* »[26]. En

22 « *A number of terms have come into use for parts of an operating system. The term `supervisory program' has already appeared above. The supervisor is the program which maintains ultimate control of the machine at all times and to which control reverts when a routine finishes its function or when an unexpected stop or 'trap' occurs. Terms which are used more or less synonymously with 'supervisor' are 'executive routine,' 'monitor,' 'master control routine'* » (Orchard-Hays, 1961, p. 290).

23 L'utilisation d'une partie pour désigner l'ensemble est connue en linguistique comme *pars pro toto* ou métonymie.

24 709 renvoie à la machine IBM 709 pour laquelle le système fut développé.

25 « *The initial problem facing the committee was to define what was meant by a 'system'* » (Shell, 1959, p. 124).

26 « [it] *coordinates the use of the various parts of the SHARE 709 System and is responsible for maintaining the computer in continuous operation during the processing of a group of independent jobs* » (Bratman & Boldt, 1959, p. 152).

outre, le système a un format standardisé pour les tâches que la machine peut facilement interpréter, cette standardisation réduisant sérieusement le temps perdu entre deux tâches. C'est une description qui correspond parfaitement à un système de traitement par lots, mais ni le mot « lot » ni le mot « système d'exploitation » ne sont utilisés.

Dans le manuel de la communauté SHARE par contre, le système est appelé SOS, pour *SHARE Operating System*. Dans l'introduction du manuel, on peut lire :

> […] le SHARE Operating System, communément appelé SOS, est un complexe très flexible de langages, procédures et codes machine. Le système a trois objectifs : aider le programmeur du 709 pendant la préparation des programmes ; alléger l'opérateur des tâches qui peuvent être automatisées ; et rendre l'installation informatique plus efficace et mieux documentée[27].

SOS est en réalité « un système intégré » qui contient trois sous-systèmes : le compilateur-assembler-traducteur de SHARE (SCAT) ; le système pour le *debugging* ; et le moniteur.

La nomenclature utilisée dans les publications officielles (ACM) et celle utilisée dans la pratique (la communauté SHARE) ne coïncident pas. Pourtant, il est évident que le terme *operating system* ne s'était pas encore imposé et que la définition même était encore flottante, désignant parfois le système de programmation ou même le système intégré. Cela changera avec les systèmes ultérieurs à SOS.

Le SHARE Operating System aura peu de succès dans la communauté SHARE parce que FORTRAN n'était pas bien supporté et que son langage d'instructions était trop compliqué (Akera 2001, pp. 731-733). En 1961, 76 % des installations IBM sont opérées sous le système Fortran Monitor System (FMS) au lieu de SOS (Larner 1987, p. 819). FMS avait été développé par un grand utilisateur institutionnel North American Aviation en 1959 pour faciliter l'usage de FORTRAN. FMS était « *un moniteur, un programme superviseur pour 709/7090 FORTRAN, FAP*[28] *et les programmes objet* »[29]. Si SOS avait été conçu comme un outil de programmation, FMS était dès le début un chargeur et connecteur (*loader & linker*) pour des programmes en FORTRAN. Peut-être que le succès de FORTRAN et son profil de « *langage*

27 « […] *the SHARE operating system, familiarly known as SOS, is a highly flexible complex of languages, procedures and machine codes. The threefold purpose of the System is to provide assistance to the 709 programmer in the coding and check-out phase of program preparation, to assume from the 709 machine operator those burdens that may be sensibly automated and to provide the computer installation with an efficient operation and complete and accurate records on machine usage* » (Homan and Swindle, 1959, sec. 01.01.01).

28 FAP est le macro-assembleur du IBM 709/7090.

29 « […] *the Monitor is a supervisory program for 709/7090 FORTRAN, FAP, and object programs* » (Reference Guide, 1961, p. 61).

de programmation »[30] a rendu lentement possible la distinction claire entre le système de programmation et le système d'exploitation (même si cette séparation est parfois artificielle et problématique). Par l'intégration du langage FORTRAN dans un environnement comme FMS qui facilite l'écriture, la compilation et la combinaison de programmes, une distinction entre langage et système (d'exploitation) devient visible pour l'utilisateur.

Cette évolution est explicitée par George H. Mealy qui faisait partie du groupe de la RAND ayant amélioré SOS pour intégrer FORTRAN, le résultat étant le RAND-SHARE Operating System (1962). Dans son rapport sur les *Operating Systems*, Mealy écrivait :

> On a une machine pour exécuter des tâches, pas pour programmer des systèmes. On appelle souvent les systèmes entre le programmeur et la machine des « systèmes de programmation », mais cela met trop en exergue les auxiliaires mécaniques de codage et ne met pas en valeur d'autres aspects de l'opération. Par *operating systems* on désigne l'ensemble des outils de programmation, debugging et outils opérationnels avec lequel travaille le programmeur[31].

Ainsi, *operating system* commence à englober et à contrôler de plus en plus les systèmes de programmation de l'ordinateur. Dans la même ligne, Bob Bemer d'IBM considérait les *operating systems* comme la Phase III dans le développement des systèmes de programmation[32], l'*operating system* englobant littéralement les systèmes de programmation (Bemer 1962).

Une séparation semblable entre système de programmation et *operating system* se trouve dans l'introduction de manuel du RAND-SHARE Operating System :

> Un *operating system* est un complexe de routines d'ordinateur qui sont utilisées pour faire entrer et sortir les programmes et les données de la machine, pour transformer des données (incluant la compilation et l'assemblage de programmes), superviser des tâches et les mettre en

30 On parle d'abord plutôt de systèmes de programmation, le fait de parler de *langage* de programmation est apparemment né dans les communautés des utilisateurs, en particulier USE (1955). L'emphase sur le langage a probablement aidé à fortifier l'idée qu'un système de programmation peut être développé indépendamment d'une machine et peut être d'application universelle, voir (Nofre, Priestley & Alberts, 2014). FORTRAN par exemple est un système qui comporte deux éléments, le langage et le traducteur qui adapte le code à la machine.

31 « *The object of having a machine is to run jobs, not programming systems. To call the systems that stand between the programmer and the machine « programming systems » is to place undue emphasis on mechanical coding aids and not enough emphasis on the other aspects of operation. By 'operating systems' we shall mean the whole complex of programming, debugging and operational aids with which the programmer deals* » (Mealy, 1962, p. 4).

32 Pour Bemer, la Phase I était la programmation de l'ordinateur sans aide, la Phase II la programmation partiellement automatisée (conversion, chargement, connexions, assemblage et compilation), et en Phase III finalement, le système d'exploitation prend en charge plusieurs tâches comme par exemple la communication avec les périphériques entrée et sortie, etc.

séquence, et pour faciliter la communication entre le programmeur et les éléments du système d'exploitation[33].

Comme pour SOS, le RAND-SHARE Operating System vise trois choses : l'efficacité dans l'utilisation du temps de la machine ; l'efficacité dans l'opération de la machine ; et l'efficacité dans la programmation. Sauf que, désormais, l'opérateur n'est même plus mentionné, et l'*operating system* commence à gouverner les systèmes de programmation. Cette évolution remplace l'opérateur par l'*operating system* comme interface principale entre programmeur et machine (du moins, en théorie, mais certainement pas dans la pratique !).

Cette tendance s'approfondit avec l'*operating system* développé par IBM, IBSYS.

Le système d'exploitation 7090/7094 IBSYS est constitué d'un ensemble intégré de programmes-systèmes qui opèrent sous le contrôle et la coordination du *System Monitor*. Le *System Monitor*, en coordonnant l'opération des sous-systèmes, permet qu'une séquence de tâches sans relation entre elles peut être exécutée sans, ou presque sans, l'intervention d'un opérateur. En réduisant le degré de participation humaine dans la mécanique du traitement d'information, le 7090/7094 IBSYS Operating System garantit que les tâches sont traitées plus vite, plus efficacement et avec moins d'erreurs humaines[34].

Cette description efface littéralement l'opérateur de l'équation et met l'*operating system* à sa place, comme catalyseur pour le programmeur. Le système contrôle et réduit les problèmes qui peuvent apparaître en traitement d'informations et qui sont dus à l'homme. L'*operating system*, en particulier le moniteur, est mis en tête de la hiérarchie des programmes et contrôle la configuration de l'ordinateur, ses périphériques et ses utilisateurs. Et avec le nom même d'*operating system*, la philosophie opérationnelle s'impose de plus en plus.

33 « *An operating system is a complex of computer routines which are used to get programs and data into and out of the machine, transform data (including program assembly and compilation), supervise job and task sequencing, and facilitate the communication between the programmer and components of the operating system* » (Bryan, 1962, p. III).

34 « *The 7090/7094 IBSYS Operating System consists of an integrated set of system programs operating under the executive control and coordination of the System Monitor. The System Monitor, by coordinating the operation of the subsystems, allows a series of unrelated jobs to be processed with little or no operator intervention. By reducing the degree of human participation in the mechanics of data processing, the 7090/7094 IBSYS Operating System ensures that jobs are processed faster, more efficiently, and with less likelihood of human error* » (IBSYS, 1964, p. 3).

Conclusion

Le développement fulgurant du logiciel va de pair avec les avancées technologiques en matière de mémoire de stockage et de vitesse de processeur. Le développement des programmes est d'abord porté par les utilisateurs, mais vers 1960 une nouvelle industrie, celle du logiciel, naît. Avec la croissance en quantité et en complexité des programmes, il devient nécessaire qu'un ordinateur soit accompagné d'une sorte de système d'exploitation. Depuis la fin des années 1950, l'utilisateur humain qui calculait déjà plus lentement que l'ordinateur, est aussi dépassé par les périphériques qui lisent et écrivent des données beaucoup plus vite que lui. Du lecteur de cartes perforées fonctionnant à deux ou trois cartes par seconde, on passe à des bandes magnétiques dont la vitesse de lecture atteint plus de 10 000 cartes par seconde. En outre, avec FORTRAN, l'ordinateur « écrit » (c'est-à-dire qu'il assemble, compile et exécute) un programme plus vite qu'un utilisateur humain, une fois que l'*automatic programming* ou *programming programs* sont devenus communs. Tous ces éléments conduisent à la naissance des systèmes d'exploitation.

Plusieurs philosophies coexistent, pour ne pas dire que chaque ordinateur ou chaque entreprise suit sa propre philosophie, incarnée dans son propre système d'exploitation. Toutefois, une philosophie emporte la majorité des suffrages, la philosophie opérationnelle sous-entendue avec le terme *operating system*. Cette vision s'appuie sur l'automatisation de l'opérateur humain et est étroitement liée au traitement par lots (*batch processing*). Elle est surtout portée par IBM qui domine le marché des ordinateurs. En quelque sorte, cette philosophie est la poursuite de la stratégie commerciale IBM consistant à louer les ordinateurs et à séparer l'utilisateur de la machine par l'intermédiaire des opérateurs, qui, sous forme automatisée, deviennent l'*operating system*[35].

Avec l'avènement de la multiprogrammation et du *time-sharing* la variété et la complexité des systèmes d'exploitation augmentera énormément. Même si désormais beaucoup de systèmes vont assumer des fonctionnalités qui sont étrangères à l'ancien opérateur humain, comme la segmentation, l'ordonnancement etc., le terme *operating system* va rester et devenir le terme utilisé par tout le monde.

35 Évidemment, la pratique ne correspond pas complètement à cette vision automatisée qui repose souvent sur des métaphores. L'opérateur humain reste nécessaire, mais son rôle change : il devient plutôt assistant ou administrateur du système d'exploitation.

Bibliographie

Adams C.W., Gill, S. & *al.* (eds.) (1954). *Digital Computers: Business Applications.* Summer program.

Adams C.W. (1955). « Developments in programming research ». *AIEE-IRE 1955 Eastern joint AIEE-IRE computer conference proceedings*, pp. 75-79.

Adams C.W. (1987). « A batch-processing operating system for the Whirlwind I computer ». *AFIPS Conference Proceedings,* vol. 56, pp. 785-789.

Akera, A. (2001). « Voluntarism and the fruits of collaboration: The IBM user group Share ». *Technology and Culture*, 42(4), pp. 710-736.

Bauer W.F. (1956). « An Integrated Computation System for the ERA-1103 ». *ACM,* 3 (3), pp. 181-185.

Bauer W.F. & West G.P. (1957). « A system for general-purpose digital-analog computation ». *ACM*, 4 (1), pp. 12-17.

Bauer W.F. (1956). « Use of Automatic Programming ». *Computers and Automation*, 5 (11), pp. 6-11.

Bauer W.F. (1958). « Computer Design from the Programmer's Viewpoint ». *Proceedings Eastern Joint Computer Conference*, December 1958, pp. 46-51.

Bauer W.F. & Rosenberg A.M. (1972). « Software – Historical perspectives and current trends ». *Fall Joint Computer Conference,* 1972, pp. 993-1007.

Bemer R. (1962). « The Present Status, Achievement and Trends of Programming for Commercial Data Processing ». *In* Hoffmann (dir.) *Digitale Informationswandler*, Wiesbaden, pp. 312-349.

Bennington H.D. & Gaudette C.H. (1956). « Lincoln Laboratory Utility Program System ». *AIEE-IRE 1956 joint ACM-AIEE-IRE western computer conference*, p. 21.

Bratman H. & Boldt I.V. (1959). « The SHARE 709 System: Supervisory Control ». *Communications of the ACM*, 6 (2), pp. 152-155.

Breheim D.J. (1961). « 'Open Shop' Programming at Rocketdyne Speeds Research and Production ». *Computers and Automation,* 10 (7), pp. 8-9.

Brinch Hansen P. (2001). *Classic Operating Systems: From Batch Processing to Distributed Systems.* Berlin : Springer.

Bryan G.E. (1962). *The RAND Share operating system manual for the IBM 7090.* Memorandum RM-3327-PR, Santa Monica, CA.

Campbell-Kelly M. (2003). *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry.* Cambridge, MA : MIT Press.

Ceruzzi P. (2003). *A history of modern computing* (2nd ed.). MIT Press, Cambridge, Massachussetts.

Culler G.J. & Fried, B.D. (1963). *An Online Computing Center for Scientific Problems*. M19-3U3, TRW report.

Dines R.S. (1966). « Telecommunications and supervisory control programs ». *Computers and Automation* 15 (5), pp. 22-24.

Drummond R.E. (1987). « BESYS revisited ». *AFIPS Conference Proceedings,* vol. 56, pp. 805-814.

Fisher F.P. & Swindle G.F. (1964). *Computer programming systems.* New York : Holt, Rinehart and Winston.

Frank W.L. (1956). « Organization of a Program Library for a Digital Computer

Center ». *Computers and Automation,* 5 (3), pp. 6-8.

Grabbe E.N., Ramo S. & Woolridge D.E. (1959). *Handbook of Automation, Computation and Control,* volume II. New York : Wiley.

Hassitt, A. (1967). *Programming and Computer systems.* New York & London : Academic Press.

Homan C.E. & Swindle G.F. (1959). *Programmer's Manual for the SHARE Operating system*. IBM.

Hopper G. (1954). *ACM Glossary.* ACM.

Reference Guide to the 709/7090 FORTRAN Programming System (1961) (includes material from IBM 709/7090 FORTRAN Monitor, form C28-6065). IBM : Poughkeepsie.

IBM 7090/7094 IBSYS system operator's guide (1964). Poughkeepsie : IBM.

Knuth D.E. & Pardo L. (1979) « The early development of programming languages ». *In* J. Belzer, A.G. Holzman & A. Kent (dir.). *Encyclopedia of Computer Science and Technology.* Marcel Dekker : New York, pp. 419-496.

Krakowiak S. (2014). « Les débuts d'une approche scientifique des systèmes d'exploitation ». *Interstices,* Février.

Krakowiak S. & Mossière J. (2013). « La naissance des systèmes d'exploitation ». *Interstices,* Avril.

Larner R.A. (1987). « FMS: The IBM FORTRAN Monitor System ». *AFIPS Conference Proceedings,* vol. 56, pp. 815-820.

Mealy G.H. (1962). *Operating Systems.* RAND Report P-2584. Republié partiellement in Rosen 1967.

Clark W.A., Mealy G.H. & Witt B.I (1966). « The functional structure of OS/360 ». *IBM Systems Journal*, 5 (1), pp. 3-51.

Mealy G.H. (1987). « Some threads in the development of early operating systems ». *AFIPS Conference Proceedings,* vol. 56, pp. 779-784.

Mock O.R. (1987). « The North American 701 Monitor ». *AFIPS Conference Proceedings,* vol. 56, pp. 791-795.

Moncreiff B. (1956). « An automatic supervisor for the IBM 702 ». *AIEE-IRE 1956 Joint ACM-AIEE-IRE western computer conference*, pp. 21-25.

Nelson E. (1969). « Computer Installation at TRW systems – Some Experiences and Lessons ». *Computers and Automation* 18 (8), pp. 21-22.

Nofre D., Priestley M. & Alberts G. (2014). « When Technology Became Language: The Origins of the Linguistic Conception of Computer Programming, 1950-1960 ». *Technology and Culture*, 55(1), pp. 40-75.

Orchard-Hays W. (1961). « The Evolution of Programming Systems ». *Proceedings of the IRE*, 49 (1), pp. 283-295.

Patrick, R.L. (1987). « General Motors/North American Monitor for the IBM 704 computer ». *AFIPS Conference Proceedings,* vol. 56, pp. 796-803.

Rosen S. (1967). *Programming Systems and Languages.* New York : McGraw-Hill.

Sackman H. (1970). *Man-Computer Problem solving.* Princeton etc. : Auerbach.

Shell D.L. (1959). « SHARE 709 system: a cooperative effort ». *Journal of the ACM*, 6 (2), pp. 123-127.

SHARE Operating System Manual, Distribution 1 to 5 (1960). Poughkeepsie, IBM.

Tanenbaum A. (2001). *Modern operating systems* (2nd ed.). Upper Saddle River, NJ : Prentice Hall.