Apports de Jean-Raymond Abrial en recherches et réalisations dans le domaine des SGBD et des langages formels à Grenoble[1]

Jean-Pierre Giraudin, IMAG, UGA Grenoble, jean-pierre.giraudin@imag.fr

1- Repères historiques grenoblois

Dans cette section il s'agit de montrer comment François Peccoud et Claude Delobel arrivent dans le contexte universitaire grenoblois au cours des années 1960 et font émerger des recherches et des formations en Systèmes d'Information, Bases de Données et Systèmes de Gestion de Base de Données. Ces recherches se concrétisent dans les années 1970 par l'apport essentiel de Jean-Raymond Abrial qui développe successivement à Grenoble le SGBD SOCRATE [Abrial et coll. 1970a], le modèle *data semantics* [Abrial 1973a] et le langage Z [Abrial 1977].

Jean Kuntzmann arrivé à Grenoble en 1945 crée en 1951 un laboratoire de mathématiques appliquées dans les combles du bâtiment de l'IPG (Institut Polytechnique de Grenoble), avenue Félix-Viallet, près de la gare. Ce laboratoire réalise des travaux pour des entreprises grenobloises et déménage en 1963 sur le domaine universitaire dans l'une des toutes premières constructions. Plus tard ce laboratoire devient en 1966 le laboratoire LA7 associé au CNRS appelé ensuite IMAG l'Institut de Mathématiques Appliquées de Grenoble, puis laboratoire d'Informatique et Mathématiques Appliquées de Grenoble.

Au sein de l'Ecole Nationale Supérieure d'Electronique, d'Hydraulique et de Radioélectricité de l'IPG Jean Kuntzmann met en place en 1957 une section spéciale d'ingénieurs mathématiciens transformée dès 1960 en section normale d'ingénieurs mathématiciens avec une première promotion de 12 étudiants diplômés en 1963. Claude Delobel et Alain Colmerauer[2] font partie de cette première promotion. En 1970 cette formation devient l'ENSIMAG (École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble).

Dans les années 1960 les nombreux échanges et travaux de Jean Kuntzmann avec les industriels grenoblois mettent en évidence que 80% des ordinateurs de l'époque (1401, 7070, Gamma 30, etc.) sont utilisés à calculer des payes, tenir des comptabilités et gérer des stocks. Au sein de son laboratoire il encourage d'une part François Peccoud (X63) à créer une équipe de recherche en informatique de gestion et d'autre part Claude Delobel (IPG63) avec son diplôme d'ingénieur en poche à faire un travail de recherche dans le domaine de la gestion d'entreprise en s'appuyant sur les mathématiques appliquées et l'informatique au sein du département Transformateurs de l'entreprise Merlin Gerin (aujourd'hui Schneider Electric).

François Peccoud démarre en 1964 au sein du laboratoire de calcul grenoblois son projet de recherche sur le thème « Comment l'introduction des ordinateurs va-t-elle transformer la gestion des entreprises ? ». Le laboratoire venait de s'équiper d'un IBM 7044 et les seuls langages de programmation disponibles étaient Cobol, Fortran et Algol. Les méthodes de conception des systèmes d'information étaient quasi inexistantes si ce n'est le début de méthodes dites d'analyse (Ariane, Corig, Minos, Protée, etc.) pour guider et standardiser le métier du développeur d'applications en informatique de gestion. François Peccoud va alors se plonger dans des chantiers concrets de mise en place de SI administratifs dont celui de l'étude de faisabilité d'un dossier médical informatisé pour l'hôpital de Grenoble.

[1] Ce document doit beaucoup à l'ouvrage collectif *« Du Big Data à l'IA - 60 ans d'expérience en traitement des données, des informations et des connaissances à Grenoble »* publié fin 2024 par UGA Editions à Grenoble.
[2] Alain Colmerauer et Philippe Roussel créent le langage PROLOG (PROgrammation en LOGique) en 1972.

En 1967 le centre scientifique IBM dirigé par Jean-Jacques DUBY est créé à l'IMAG avec l'installation d'un IBM 360/67. Dans le cadre du projet à l'hôpital de Grenoble, sur la demande de François Peccoud et la recommandation de Louis Bolliet, Jean-Raymond Abrial (X58) arrive à Grenoble fin 1969 et apporte alors une contribution essentielle pour combiner une nouvelle vision du dossier médical des patients avec l'approche Bases de Données et un langage d'interrogation de données. La figure 1 construite à partir d'une photo[3] de l'organigramme de l'IMAG visualise le cadre scientifique et matériel que trouve Jean-Raymond Abrial à son arrivée à Grenoble. L'IMAG comptait sept groupes de recherches dont celui de François Peccoud sous le titre *Gestion*, et des services de gestion (des ordinateurs (360/67, …), des calculateurs analogiques, des langages de programmation et des systèmes (CP/CMS, …), de mécanographie, etc.).

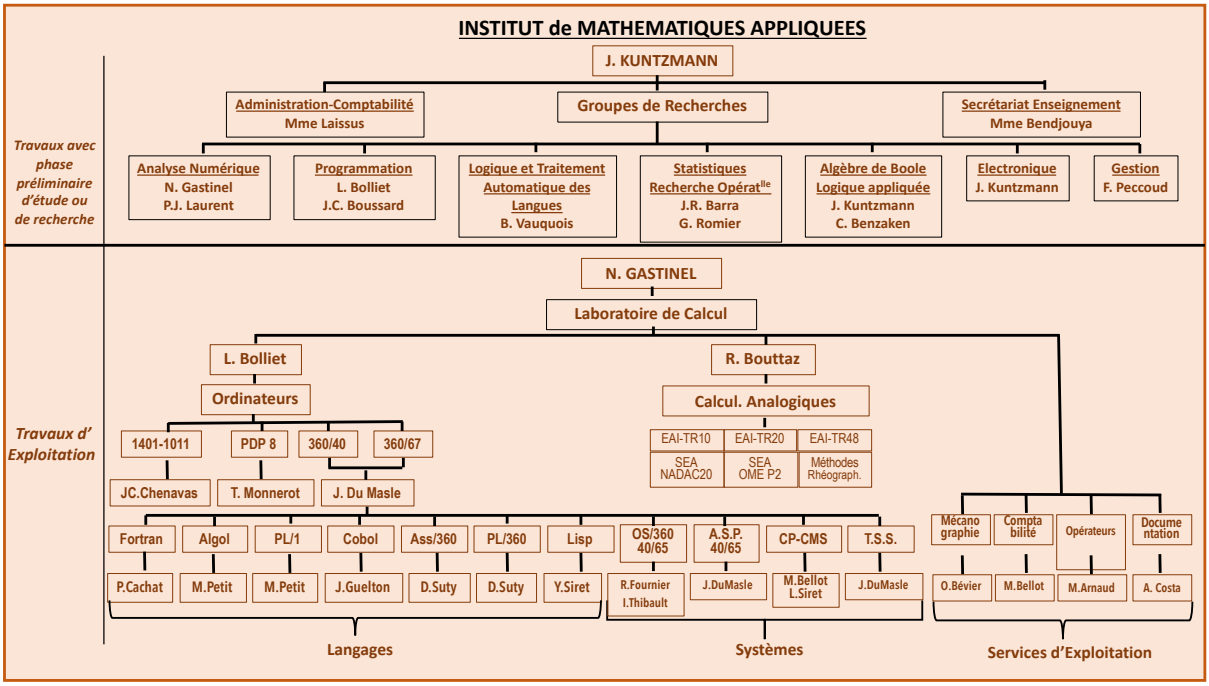


Figure 1 - Organigramme de l'IMAG vers 1968-1970

Après le constat par François Peccoud à l'hôpital de Grenoble de l'impossibilité de gérer et interroger rapidement des données et des dossiers pour des usages variés avec les seuls outils classiques de manipulation de fichiers et le langage Cobol, c'est sur cet ordinateur IBM 360/67 que Jean-Raymond Abrial et sa petite équipe d'ingénieurs vont programmer en langage assembleur, en quelques mois au début des années 1970, le premier prototype du SGBD SOCRATE[4], le premier Système de Gestion de Bases de Données français. À partir de 1972, après la réalisation deux prototypes de SOCRATE, Jean-Raymond Abrial infléchit ses recherches en quittant le champ strict des SGBD pour approfondir les liens sémantiques entre les données. Il définit en 1973 le langage de spécification *data semantics* comme un **graphe de navigation entre les données**. Il poursuit ces travaux à Grenoble pour initialiser le langage formel Z (Z0, Z1, …) en 1976-1977, travaux qu'il continue à EDF et à Oxford. Ces travaux se parachèvent au milieu des années 1980 avec B à Cambridge, un modèle de spécification des

[3] La photo originale un peu floue de l'organigramme de l'IMAG est accessible sur le site Web de l'association ACONIT à l'URL https://aconit.inria.fr/omeka/items/show/687.html

[4] La phrase *« En 1968, il a réalisé le système de base de données SOCRATE »* se trouve dans une biographie wikipedia de Jean-Raymond Abrial. Cette date erronée provient d'une erreur dans un ouvrage. Jean-Raymond Abrial n'arrive à Grenoble qu'en 1969, constitue son équipe d'ingénieurs et réalise le travail de spécification et de prototypage en 1970.

applications et un outil opérationnel qu'il applique concrètement pour le développement à la RATP des parties critiques du métro sans conducteur pour la nouvelle ligne 14. Le langage, la démarche et l'outil B sont utilisés par la suite dans nombreux autres projets de métros et navettes sans conducteurs tant en France qu'à l'étranger. Dans ce document nous nous limitons aux langages *data semantics*, Z0 et Z1. Nous ne traitons pas les concepts de machine abstraite et de raffinement qu'il a développés dans B, Event-B et la plate-forme Rodin.

2- Prototype du SGBD SOCRATE

2.1- Contexte local

L'environnement IMAG, l'engagement de François Peccoud dans le projet à l'hôpital et les idées innovantes de Jean-Raymond Abrial vont faciliter le financement d'un projet de SGBD par des contrats de l'IRIA[5] et de la DGRST[6]. Un groupe de 6 à 8 chercheurs et ingénieurs est constitué sous la responsabilité de Jean-Raymond Abrial. Il aboutit, en moins de deux ans, au premier prototype du SGBD SOCRATE. Ce résultat doit beaucoup à une gestion sans faille du projet et à un principe appliqué avec rigueur : **spécifier avant de réaliser**. Jean-Raymond Abrial réunit régulièrement son équipe pour arrêter des spécifications et interdit tout développement individuel avant la validation collective des spécifications. Ce travail de recherche novateur est consacré par une présentation collective en 1970 de trois thèses [Beaume 1970, Morin 1970, Vigliano 1970] comme extensions du document fondateur « Projet SOCRATE (1) – Spécifications générales » [Abrial et coll. 1970a]. Les originalités de SOCRATE sont nombreuses et d'autres thèses sont publiées ensuite sur des évolutions conceptuelles et architecturales [Mazaré 1973, Barbosa 1975, Portal 1975].

2.2- Contexte général

Les fortes évolutions en hardware et software durant les années 1960 avec l'arrivée des mémoires secondaires à accès aléatoire (disques, tambours) et des systèmes à accès multiples vont rendre les applications en informatique de gestion à base de fichiers et de programmes Cobol (langage normalisé en 1959) inadaptées à la bonne marche des organisations. Ces organisations souhaitent que les opérations de base sur les données (création, mise à jour, suppression, interrogation) soient faciles, rapides et exécutables en permanence par un personnel non spécialisé en informatique. Les débuts des SGBD IDS de General Electric en 1964 et IMS d'IBM en 1966, du modèle de données CODASYL en 1969 et du modèle relationnel d'Edgard Codd en 1970 sont autant de pistes différentes pour de nouvelle solutions de conception et développement en informatique de gestion. En 1970 il n'y a pas de SGBD relationnels. IMS est de type hiérarchique et IDS de type réseau et les langages de programmation et d'interrogation sont encore affaire de spécialistes. SQUARE l'ancêtre de SQL n'apparaît qu'en 1975 tout comme QBE (Query By Example) pour une interrogation par interface graphique et l'architecture ANSI/SPARC de Charles Bachman pour distinguer 3 niveaux de modélisation pour les données (externe, conceptuel ou logique, interne ou physique).

2.3- Structure de données SOCRATE

En matière de gestion de données, SOCRATE combine une structuration hiérarchique type Cobol et l'introduction de nouveaux liens de type réseau entre les informations : référence (pointeur vers une donnée), inverse (ensemble de pointeurs vers un sous-ensemble) et anneau (liste chaînée fermée entre données). La figure 2, inspirée de la thèse de Georges Vigliano,

[5] L'IRIA (Institut de Recherche en Informatique et Automatique) créé le 3 janvier 1967 dans le cadre du Plan Calcul devient l'INRIA (Institut National de Recherche en Informatique et en Automatique) le 1er janvier 1980.
[6] DGRST : Délégation Générale à la Recherche Scientifique et Technique

décrit un exemple de structure de données SOCRATE pour une application hospitalière avec des constructeurs et des types structurels : a) l'entité PERSONNE pour regrouper les données de chaque personne, b) l'inverse MALADES pour décrire un sous-ensemble (les malades), c) l'anneau POSTES pour matérialiser des liaisons inverses l'une de l'autre (pour connaître les postes d'un service et le service d'un poste), d) la structure conditionnelle avec l'attribut NOMJEUNEFILLE, e) la structure hiérarchique EXAMENS (les examens lors d'un séjour d'un malade).

```
DEBUT
    ENTITE 10 000 PERSONNE
    DEBUT   NOM MOT 30
            PRENOM MOT 30
            SEXE (MASCULIN FEMININ)
            ETATCIVIL (CELIBATAIRE MARIE)
            SI SEXE = 'FEMININ' et ETATCIVIL = 'MARIE'
            ALORS NOMJEUNEFILLE MOT 30
    FIN

    MALADES INVERSE TOUTE PERSONNE

    ENTITE 50 000 SEJOUR
    DEBUT   MALADE REFERE UNE PERSONNE
            DATE-ENTREE
            DEBUT   JOUR DE 1 A 31
                    MOIS DE 1 A 12
                    ANNEE DE 1960 A 2000
            FIN
            DATE-SORTIE IDEM DATE-ENTREE
            SERVICEACCUEIL REFERE UN SERVICE
            ENTITE 10 EXAMEN
            DEBUT   TYPE MOT 20
                    RESULTAT TEXTE
            FIN
    FIN

    ENTITE 100 SERVICE
    DEBUT   NOM MOT 30
            CHEF REFERE UNE PERSONNE
            POSTES ANNEAU
    FIN

    ENTITE 5000 POSTE
    DEBUT   CODE DE 1 A 9999
            CATEGORIE MOT 4
            SERVICE REFERE POSTES DE UN SERVICE
            OCCUPANT REFERE UNE PERSONNE
    FIN
FIN
```

Figure 2 - Définition d'une structure simplifiée de données hospitalières

2.4- Programmation SOCRATE

Un langage simple de création, mise à jour et d'accès aux données est défini. Des commandes sont utilisées pour « peupler » et interroger une base de données : G (génération), T (suppression), M (mise à jour), I (interrogation) et N (dénombrement). Les utilisateurs ne se préoccupent pas des mécanismes de stockage en mémoires physiques. La figure 3 énumère quelques instructions de peuplement, de mise à jour et d'interrogation de la base de données de la figure 2.

```
FAIRE
/* BOUCLE SUR LA CREATION DE PERSONNES */
/* UTILISATION DES VARIABLES Z1 et Z2 POUR SAISIR ET VERIFIER NOM ET PRENOM */
    I 'NOM, PRENOM DE LA PERSONNE ?'
    M Z1 = EXT
    M Z2 = EXT
    SI EXISTE UNE PERSONNE AYANT NOM = Z1 ET PRENOM = Z2
        ALORS I 'PERSONNE DEJA DEFINIE REFAIRE'
        REFAIRE
    FIN
/* CREER LA PERSONNE ET METTRE A JOUR SES CARACTERISTIQUES */
    G UNE PERSONNE X1
    M NOM DE X1 = Z1
    M PRENOM DE X1 = Z2
    M SEXE DE X1 = EXT
    M ETATCIVIL DE X1 = EXT
    SI SEXE = 'FEMININ' et ETATCIVIL = 'MARIE'
        ALORS M NOMJEUNEFILLE = EXT
    FIN
/* FIN DE CREATION DE LA PERSONNE */
…
/* SI LA PERSONNE X1 EST MALADE */
/* CREER UN SEJOUR */
    G UN SEJOUR X2
/* ETABLIR LA RELATION ENTRE SEJOUR ET MALADE ET METTRE A JOUR MALADES */
    M MALADE DE X2 = X1
    G UN MALADES = X1
/* METTRE A JOUR LES AUTRES INFORMATIONS DE SEJOUR … */
…
REFAIRE
FIN
…
/* COMPTER LES MALADES ET AFFICHER LES NOMS ET PRENOMS DES MALADES */
    M Y1 = N TOUT MALADES
    I Y1
    I 'MALADES'
    POUR TOUT MALADES
        I NOM
        I PRENOM
    FIN
…
```

Figure 3 - Instructions de gestion de données hospitalières

2.4- Caractéristiques du prototype

Jean-Raymond Abrial combine avec son équipe trois idées très novatrices pour un système de Bases de Données.

Modèle réseau **–** La première idée c'est le choix d'un modèle permettant de disposer à la fois des caractéristiques d'un modèle hiérarchisé (comme les fichiers Cobol ou le SGBD IMS d'IBM) et d'un modèle réseau comme celui de CODASYL. Cette originalité est sa force mais aussi sa faiblesse car le modèle SOCRATE n'est pas compatible avec la norme CODASYL proposée par un comité composé principalement de membres anglo-saxons.

Espace virtuel **-** La deuxième idée est relative à la dissociation entre un espace virtuel des données et un espace physique de stockage sur disque. L'espace virtuel est aussi gigantesque que l'on veut, alors que l'espace physique est contraint par la taille des unités de disque. Toute information pouvant exister a initialement une adresse virtuelle. Pour le premier prototype la

taille de la mémoire virtuelle est de 2^{32} mots de 32 bits, une adresse virtuelle est donc codée sur un mot de 32 bits. L'espace réel et l'espace virtuel sont découpés en pages (blocs de n mots consécutifs). La figure 4 illustre la projection de l'espace virtuel sur l'espace réel qui se fait par hash-code sur les adresses de pages. Les collisions d'adresses virtuelles différentes sur la même page réelle sont gérées par des mots de chaînage. Les chaînages nécessitent des entrées-sorties supplémentaires qui dégradent les temps de réponse, ce qui explique que chaque page de l'espace virtuel est découpée en 32 sous-pages de 7 mots pour limiter la taille des échanges entre la mémoire principale et la mémoire secondaire. Une sous-page de l'espace réel fait 8 mots dont 1 mot de chaînage. Une des difficultés réside dans le choix d'une bonne taille de pages et sous-pages selon les applications et la taille de la mémoire secondaire. Pour l'époque, cette idée est révolutionnaire pour le développement d'un SGBD, mais elle est trop en avance par rapport à la technologie et aux performances des disques disponibles. Par la suite, un processeur hardware spécialisé est développé qui se substitue à du logiciel pour accélérer le processus d'accès.

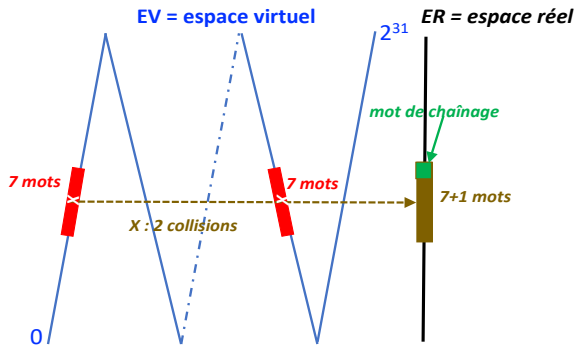


Figure 4 – Projection de l'espace virtuel sur l'espace réel

Langage de requêtes - La troisième idée est l'introduction d'un langage de requêtes structurées, comme l'a été par la suite le langage SQL (Structured Query Language) pour le modèle relationnel. La grande différence entre le langage d'interrogation de SOCRATE et celui de SQL, c'est que SOCRATE repose sur le modèle réseau, alors que le langage SQL repose sur le modèle relationnel. Avec SOCRATE, il est nécessaire de dire comment le résultat est construit en navigant au travers de la structure de données en réseau. Les programmes écrits en SOCRATE s'appuient sur les approches algorithmiques introduites par Algol. Quelques exemples de requêtes sont donnés dans la figure 3.

2.5- Utilisations de SOCRATE

L'épopée scientifique et industrielle de SOCRATE est largement développée dans le mémoire de master en sciences humaines et sociales de Jean Ricodeau [Ricodeau 2016], par Claude Delobel dans le chapitre III.1 de l'ouvrage collectif *« Du Big Data à l'IA - 60 ans d'expérience en traitement des données, des informations et des connaissances à Grenoble »* [Adiba et coll. 2024] ainsi que dans le document[7] publié sur le site de l'association ACONIT en 2021 par Christian Jullien ancien chef de projet SOCRATE et expert SGBD à SYSECA. L'année 1963 mentionnée dans le titre du mémoire de Jean Ricodeau n'indique pas le début de SOCRATE mais simplement le début des recherches grenobloises en informatique de gestion par François Peccoud à l'IMAG. Ces recherches vont révéler en particulier sur le projet de gestion de données hospitalières la nécessité de nouvelles approches pour des mises à disposition rapides et simplifiées des données indispensables pour une efficacité des personnels soignants.

[7] https://www.aconit.org/spip/spip.php?article657

Les 2 décennies de SOCRATE sont marquées par plusieurs périodes. La première période avec la réalisation rapide du premier prototype en 1970-1971 à l'IMAG. Le second prototype est une version conversationnelle développée sur l'IBM 360/67, machine du centre scientifique IBM grenoblois dotée du système CP/CMS. Cette version a été ensuite utilisée dans de très nombreux travaux grenoblois dans les années 1970-1980 : projets et de thèses d'une grande variété comme l'utilisation relationnelle d'une base de données SOCRATE, l'extension SOCRATE pour gérer des documents structurés, le prototypage d'applications de gestion avec SOCRATE, la gestion de généalogies, etc.

Fin 1971 les travaux de Jean-Raymond Abrial à l'IMAG sur le SGBD SOCRATE commencent à être connus (publications, thèses, etc.) dans les services informatiques des administrations et des entreprises. Les chercheurs de l'équipe SOCRATE rejoignent alors la société de services ECA Automation pour développer une version industrielle du SGBD sur IRIS 50 pour le compte de la Sécurité Sociale. Cette première version industrielle de SOCRATE est développée en 2 ans sous la direction de Stefen Stépanian. Dès 1974 de nombreuses applications opérationnelles sont réalisées comme à la Gendarmerie pour la consultation d'une base données de personnes lors de contrôles, à l'hôpital de Grenoble pour la gestion du dossier médical (projet initialisé fin 1969 et fondateur de SOCRATE), etc. De nouvelles versions de SOCRATE sont développées en langage machine sur des ordinateurs IBM et Siemens pour des applications plus ambitieuses que la gestion administrative : système de commandement de la Marine nationale, poste de commande de sites nucléaires, etc.

Dès les années 1980 une « souche portable » de SOCRATE est réalisée basée sur le développement d'un langage de macro-génération pour l'écriture des composants du SGBD. L'ouverture sur une gamme de plus en plus large de machines et d'OS (mini-ordinateurs SOLAR de la Télémécanique, DEC VAX, Bull GCOS, etc.) permet une diffusion du produit qui s'appellera CLIO et ECA Automation est renommée SYSECA en 1983.

3- Langages formels : *data semantics* et Z

3.1- Contexte local et contexte général

Dès les prototypes SOCRATE terminés à l'IMAG en 1972, Jean-Raymond Abrial approfondit les liens sémantiques entre les données. Claude Delobel revient en 1972 à Grenoble après une année passée au sein de l'équipe de Mike Senko au laboratoire d'IBM à San José où il a travaillé sur les dépendances fonctionnelles en adéquation avec les formes normales introduites par Ted Codd. Le langage de programmation logique PROLOG (PROgrammation en LOGique) est créé vers 1972 à Marseille par Alain Colmerauer (diplômé de l'ENSIMAG la même année que Claude Delobel en 1963) et Philippe Roussel. Le groupe CODASYL en 1969 propose une première normalisation des modèles de données en réseau. Ted Codd publie en 1970 l'article fondateur sur le modèle relationnel. Les publications du laboratoire d'intelligence artificielle du MIT se multiplient tant théoriques qu'orientées production industrielle assistée par robot. C'est le cas en 1971 de la thèse de Carl Hewit sous la direction de Seymour Papert : *« Description and theoretical analysis (using schemata) of PLANNER : a language for proving theorems and manipulating models in a robot »*. Déjà à la recherche de la spécification idéale, tous ces travaux fondamentaux vont orienter Jean-Raymond Abrial à faire des choix associant simplicité et théorie, faits et règles logiques pour définir le langage de spécification *data semantics*. Ce modèle de données fait l'objet de premières présentations en 1973 à Grenoble et en école d'été. La conférence « Working conference on DataBase Management » de l'IFIP à Cargèse en Corse en 1974 donne une visibilité de Grenoble sur les recherches en bases de données et sémantique des données [Abrial 1974].

3.2- *Data semantics*

Le but est de définir la connaissance que l'on veut acquérir et stocker dans une base de données. Cette connaissance peut être définie à différents niveaux par :

a) des faits élémentaires ou affirmations : Pierre Martin a 76 ans
b) des règles simples : toute personne possède nécessairement à un instant donné un état marital soit C, M, V ou D
c) des règles plus élaborées : une personne mariée ne peut plus être célibataire dans le futur
d) des règles de calcul ou de déduction : nombre de personnes mariées ou veuves = nombre de personnes dont l'état marital est M + nombre de personnes dont l'état marital est V.

Le langage *data semantics* est utilisé pour modéliser cette connaissance qui évolue dans le temps. Le langage fait une différence entre des objets concrets et des objets abstraits. Les objets abstraits peuvent être utilisés pour des mesures quantitatives (entiers, réels, etc.), temporelles (Mois, Jour, An) ou qualitatives (M, F). L'existence des objets concrets est à affirmer au sein du monde décrit. La description (sémantique) d'un objet concret dans le modèle est donnée par les liaisons qu'il a avec d'autres objets concrets ou abstraits. Chaque liaison est libellée avec le nom de deux fonctions d'accès qui peuvent avoir de multiples valeurs (fonction d'accès multivaluée) comme personnedesexe[masculin] = {John, Peter}. Ces liaisons bidirectionnelles sont des relations binaires (relations entre deux ensembles). Les ensembles de données sont de différentes catégories : des personnes, des nombres, etc.

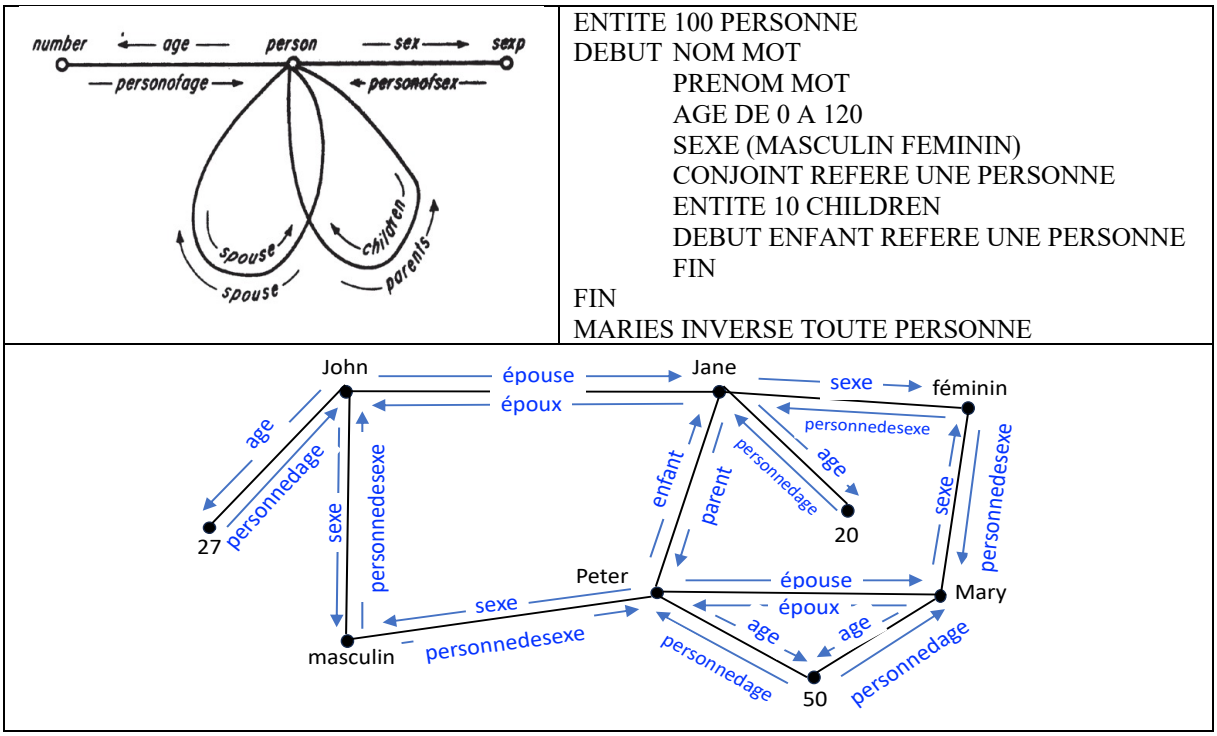


Figure 5 - Structure de données *Data semantics* et SOCRATE.

La figure 5 contient une représentation graphique d'une structure de données, ainsi que des données liées conformément à cette structure. La déclaration SOCRATE est aussi donnée pour compléter cet exemple simple. Il faut se poser ensuite le problème d'identification des objets concrets comme identifier les personnes par NOM et PRENOM ou un numéro INSEE. La description de faits de nature n-aires comme MARIAGE est possible par produit cartésien (*MARIAGE = PERSONNE x PERSONNE x LIEU x DATE*) dont la sémantique peut être précisée comme dans la figure 6 qui définit MARIAGE, un ensemble d'objets concrets décrit

avec des liaisons bidirectionnelles associant 2 personnes, 1 lieu et 1 date. LIEU et DATE peuvent être décrits comme produits cartésiens.

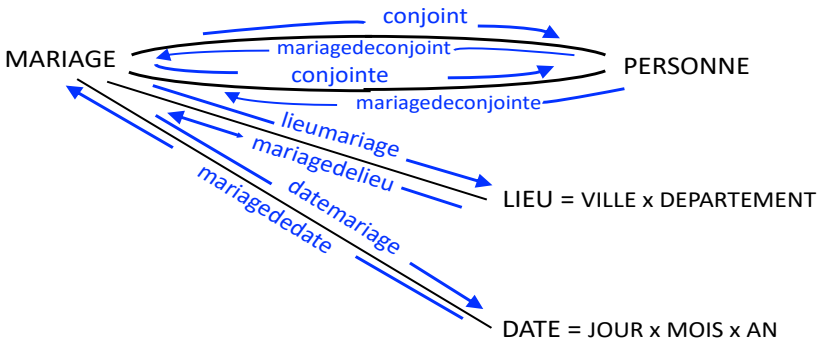


Figure 6 – Exemple d'un fait n-aire en *Data semantics*.

Des commandes permettent de créer des entités. Des notations différentes sont utilisées pour lier (←, :∋) ou délier (←, nil, :∌)des objets selon que la fonction d'accès est monovaluée ou multivaluée :

X ← GENERATE PERSONNE	nom (X) ← ABRIAL	prénom (X) ← JEAN-RAYMOND
age (X) ← 50	parent (X) :∋ Y	épouse (X):∌ Z

Le langage contient de nombreuses possibilités pour exprimer des contraintes avec les comparateurs ($= \neq < \leq > \geq$), opérateurs ensemblistes ($\in \notin \cup \cap \supset \supseteq \subset \subseteq$), quantificateurs ($\forall \exists$), et pour décrire des programmes avec des conditions, des itérations, etc. Ce langage doit être considéré comme un outil de spécification et non comme un langage de manipulation de bases de données. Jean-Raymond Abrial part de ce travail préliminaire fondé sur la théorie des ensembles et la logique des prédicats pour proposer quelques années plus tard le langage Z.

En août 1973, Jean-Raymond Abrial introduisait ses présentations du modèle sémantique de données en énumérant des qualités attendues d'un tel modèle :

a) Un modèle doit pouvoir restituer ce qu'il a appris.
b) Un modèle doit pouvoir calculer de nouvelles informations par déduction. Les déductions possibles ne doivent pas être limitées à un seul niveau et permettre des déductions de déductions.
c) Un modèle doit pouvoir évoluer car la réalité n'est pas figée.
d) Un modèle doit être performant. Il faut faire attention à l'explosion imaginative et se concentrer sur ce qui est utile et préférer ***« un modèle qui fait peu mais le fait bien ».***
e) Un modèle doit avoir la capacité de contrôler l'information qu'il acquiert. Un modèle rempli d'informations fausses est inutilisable.
f) Un modèle doit avoir la propriété de *data independance*, c'est-à-dire être utilisable en interrogation de manière indépendante par rapport au mode d'acquisition, de déduction ou de stockage des faits.
g) Un modèle doit-être capable de se décrire lui-même. Par l'autodescription, le modèle s'explique et se connaît lui-même et peut alors se contrôler. Ce n'est pas un jeu théorique dénué d'intérêt.

3.3- Langages Z0 et Z1

Le langage Z est basé sur le principe qu'une conception d'un système doit être progressive et dirigée selon des niveaux successifs. Z0 constitue le premier niveau caractérisé par une description statique du système à résoudre. Il est non algorithmique, c'est-à-dire sans instructions. Le niveau suivant Z1 introduit des instructions de type algorithmique et la notion de procédure pour exprimer la dynamique d'un système. Les présentations exhaustives de Z0

et Z1 sont données dans le manuel du langage Z (Z/13) [Abrial 1977]. Le lecteur peut trouver des approches pédagogiques de Z0 et Z1 dans le livre sur les méthodes de programmation de Bertrand Meyer et Claude Baudoin [Meyer et coll. 1978], ainsi que dans celui de Claude Delobel et Michel Adiba centré sur les bases de données [Delobel et coll. 1982].

Langage Z0

Z0 est fondé sur le langage mathématique et plus particulièrement sur le langage de la théorie des ensembles et les prédicats. Il est caractérisé par trois groupes de définition : les ensembles d'entités, les associations entre ensembles d'entités et les contraintes d'intégrité.

Les éléments de base sont les **ensembles d'entités**. Un ensemble d'entités possède un nom unique et regroupe des entités de même nature : des personnes, des enseignants, des postes, etc. Les opérations ensemblistes applicables sont : a) test d'appartenance d'une entité à un ensemble et test d'inclusion d'un ensemble dans un autre, b) opérations d'union, intersection et différence de deux ensembles, c) produit cartésien d'ensembles. Quatre modes de définition d'un ensemble d'entités sont utilisables : extension, intention, produit cartésien et calcul d'ensembles.

Extension : {célibataire, marié, voeuf, divorcé}
Intention : {x ∈ PERSONNE / ETAT_CIVIL = divorcé}
Produit cartésien : JOUR x MOIS x ANNEE
Calcul : {x ∈ PERSONNE / ETAT_CIVIL = marié}
∪ {x ∈ PERSONNE / ETAT_CIVIL = voeuf}

Z0 comme *data semantics* appartient à la classe des modèles relationnels binaires car il utilise des **associations** qui sont des relations binaires entre ensembles d'entités. Ce qui fait de lui un bon candidat pour décrire le niveau conceptuel de tout système (base de données, système d'information, programme, etc.). Une association entre deux ensembles d'entités non obligatoirement distincts est définie par deux fonctions inverses l'une de l'autre. Nous les appelons fonctions de dépendance ou fonctions sémantiques pour les distinguer des fonctions mathématiques. Chaque fonction de dépendance est définie par son nom, sa cardinalité vis-à-vis de l'ensemble source (totale ou partielle) et sa cardinalité vis-à-vis de l'ensemble cible (monovaluée ou multivaluée). Plusieurs notations sont utilisables. Dans les exemples de la figure 7 nous utilisons des notations variées proches de celle de Jean-Raymond Abrial avec des couples de cardinalités pour le premier exemple, avec des flèches simples (monovaluée) ou doubles (multivaluée) éventuellement barrées (partielle) pour les deux autres exemples. Les deux premières associations indiquent que dans le système visé toute personne est obligatoirement affectée à un service et un seul (fonction de dépendance monovaluée totale) et que dans tout service il y a 0 ou plusieurs personnes (fonction multivaluée partielle). Implicitement cette spécification précise que la création d'une personne dans le système contiendra impérativement son affectation dans un service et qu'un service pourra être créé sans personne affectée. La troisième association décrit sommairement une généalogie pour laquelle il est utile de préciser le couple des cardinalités (0, 2) qui serait par défaut (0, n) pour la fonction de dépendance *parents*. Pour la fonction de dépendance *Parents* il ne faut pas indiquer le couple de cardinalités (2, 2) qui induit une généalogie infinie des ascendants impossible à mettre à jour.

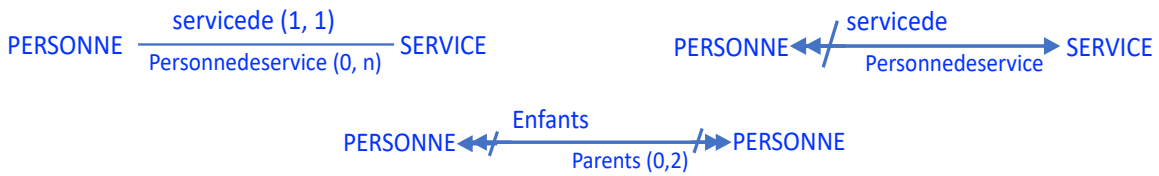


Figure 7 - Associations Z0.

Il y a 16 associations de types différents selon les cardinalités de leurs deux fonctions de dépendance. La liste ordonnée de ces associations dans la figure 8 est établie en tenant compte qu'une fonction de dépendance monovaluée totale est plus contraignante (contrainte en termes de gestion de liaisons et déliaisons entre objets) qu'une fonction de dépendance multivaluée partielle. Nous avons expérimenté l'intérêt de cet ordre dans des projets d'assistance à la conception de bases de données. Lors d'une fusion de modèles, cet ordre est utilisé pour réaliser des choix entre des associations susceptibles d'être redondantes ou substituer une association par une autre.

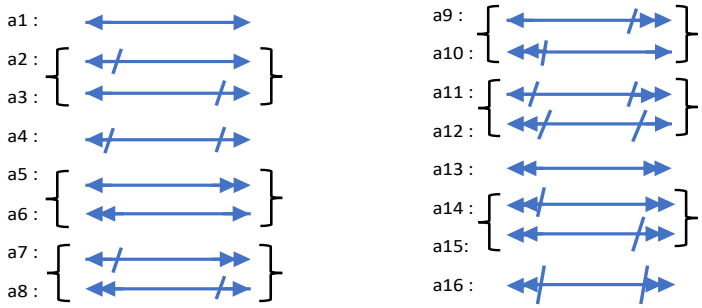


Figure 8 – 16 formes d'associations Z0.

Les notations graphiques et littérales des associations permettent de dessiner un **graphe conceptuel** du système porteur de sémantique. Diverses situations sont exprimées dans le graphe de la figure 9 : a) dans une association l'une des fonctions peut ne pas être nommée mais il est nécessaire d'en donner sa cardinalité, b) la première lettre d'une fonction monovaluée est une minuscule, et c'est une majuscule pour une fonction multivaluée, c) certains ensembles d'entités sont plus centraux ou principaux dans la description du système comme PERSONNE et PRODUIT.

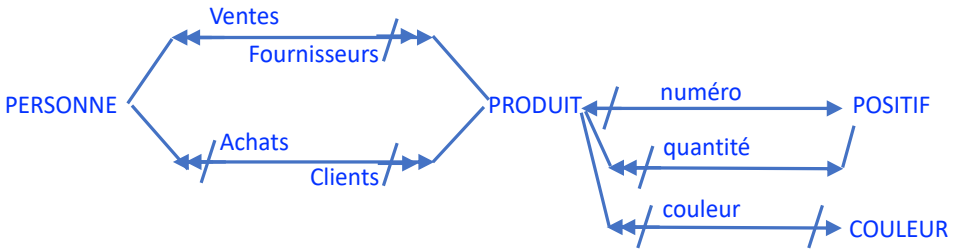


Figure 9 - Graphe conceptuel Z0.

Des expressions logiques sont utilisables pour définir des **contraintes d'intégrité** inter-ensembles et/ou inter-associations. Il est possible de comparer des entités ($=, \neq, <, \leq, >, \geq$) ou des ensembles d'entités ($=, \neq, \subset, \subseteq, \supset, \supseteq$) pour former des formules atomiques. Une formule atomique est une formule que l'on peut parenthéser, prendre la négation ($\neg$), appliquer des opérateurs logiques ($\wedge, \vee, \Rightarrow$), des opérateurs ensemblistes ($\in, \notin, \cup, \cap, \supset, \supseteq, \subset, \subseteq$) et les quantificateurs ($\forall, \exists$). De telles formules permettent d'exprimer des invariants du système, par exemple il n'y a pas de boucle dans une généalogie *($\forall p \in PERSONNE\ (Enfants\ (p) \cap Parents\ (p) = \varnothing)$)*, ou que toute personne du système est impliquée dans au moins une vente ou un achat (*$\forall p \in PERSONNE\ (Ventes\ (p) \cup Achats\ (p) \neq \varnothing)$*). Il est aussi possible d'exprimer une question au système, comme connaître les fournisseurs de produits verts :

$\{\ p \in PERSONNE \mid \exists pr \in PRODUIT\ .\ (p \in Fournisseurs\ (pr) \wedge couleur\ (pr) = \text{'vert'})\ \}$

Ces exemples Z0 montrent une continuité dans la recherche, par Jean-Raymond Abrial depuis *data semantics,* d'une **spécification précise et progressive** à l'aide du langage mathématique (universel), complétée par une représentation graphique utile pour une vision d'ensemble du

système. Cette continuité se retrouve aussi dans le langage algorithmique Z1. Le langage de manipulation et d'interrogation de données en SOCRATE était aussi algorithmique.

Langage Z1

Le langage Z1 permet de décrire des algorithmes, suites ordonnées de phrases. Chaque phrase contient des expressions Z0 (désignation d'entités, désignation d'ensembles d'entités et formulation d'expression logique). Différents types de phrase sont autorisés : création et suppression d'entités (entité ← créer (ENSEMBLE), tuer (entité)), création et suppression de liaisons fonctionnelles (fonctions monovaluées, fonctions multivaluées), affectations (variable d'entité ← entité, variable d'ENSEMBLE ← ENEMBLE), conditionnelles (si…, selon…), itératives (tantque…, répéter…, pour…), entrées-sorties (lire entité1, entité2, … ; afficher entité1, entité2, …) et procédures avec paramètres.

Le langage Z1 permet de passer d'une description statique Z0 à l'étape suivante pour s'approcher progressivement de la programmation dans un langage cible. Pour illustrer cette *algorithmisation* de Z0 nous reprenons dans la figure 10 l'exemple simple donné dans le livre de Bertrand Meyer et Claude Baudoin de la transformation d'une expression Z0 en un algorithme Z1 pour vérifier la contrainte c pour tout élément d'un ensemble A. Une liste de 209 transformations sont rassemblées dans l'annexe du document Z/14 [Abrial 1977b]. Ces transformations sont utilisées dans le document Z15 pour décrire complètement une petite application de gestion [Abrial 1977c].

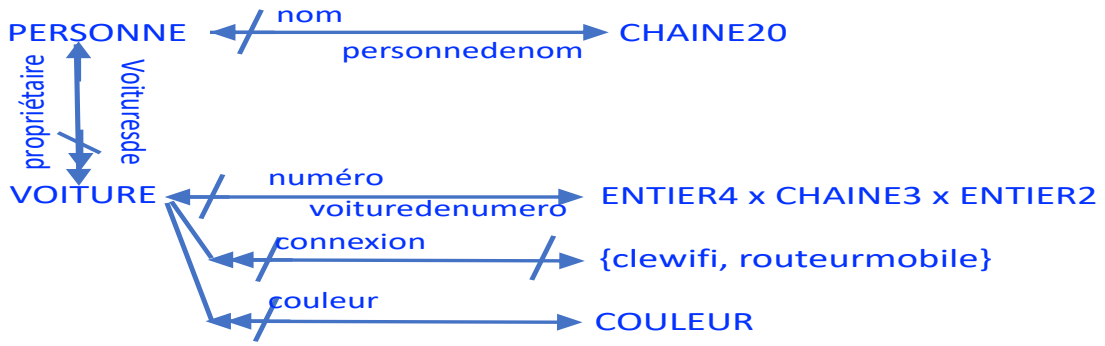
Expression logique en Z0 : $\forall x \in A \,.\, c\,(x)$

Algorithme Z1 :

```
v ← vrai
pour x∈A
        tantque v répéter
                v ← c (x)
```

Figure 10 - Transformation Z0-Z1.

Pour aider à la compréhension de ce langage nous nous limitons à lister dans la figure 11 des phrases Z1 de différents types sur un schéma Z0 simple. Le lecteur lira avec intérêt des exemples plus complets dans les livres et documents référencés. Sur le schéma Z0, nous avons appliqué plusieurs principes : utilisation de majuscules pour les noms des ensembles et pour le 1er caractère d'une fonction multivaluée, minuscules pour les fonctions monovaluées, les deux fonctions de dépendance sont nommées lorsque l'association est décrite entre deux ensembles principaux (comme PERSONNE et VOITURE), ou si elle a un rôle d'identification.



Affectations de création d'entités :

pe ← créer (PERSONNE)

v ← créer (VOITURE)

Affectations de valeurs :

nom (pe) ← 'rodin'

numéro (v) ← (1977, ZZ, 38)
connexion (v) ← 'clewifi'
couleur (v) ← 'rouge'
Affectations de liaisons fonctionnelles :
propriétaire (v) ← pe qui peut s'écrire Voiture (pe) :∃ v
Désaffectations, suppressions et modifications :
Voiture (pe) : ∄ v
tuer (propriétaire (voituredenumero (1977, ZZ, 38)))
tuer (voituredenumero (1977, ZZ, 38))
Voituresde (personnedenom ('rodin')) :∄ {v ∈ VOITURE | couleur (v) = 'rouge'}
couleur (v) ← nil

Figure 11 - Schéma Z0 et instructions Z1.

La grammaire complète du langage Z est donnée dans divers documents. D'autres opérateurs et phrases existent en Z comme l'opérateur τ : $\tau\, pe \in PERSONNE \,.\, Voiture\ (pe) = \varnothing$ désigne une personne quelconque du sous-ensemble des personnes qui n'ont pas de voitures (sorte de tirage au sort).

Démarche de conception

La démarche de conception préconisée par Jean-Raymond Abrial est fondée sur l'idée d'un premier niveau de spécification purement statique non algorithmique avec Z0 suivi de l'utilisation de Z1 pour des transformations successives pour se rapprocher progressivement du niveau des langages de programmation usuels. Si le langage de programmation utilisé dans l'étape ultime ne dispose pas du type ensemble, il est nécessaire de remplacer les ensembles par d'autres types (listes, tableaux, etc.) et d'utiliser les ordres adaptés pour les parcourir. Plus tard avec B, Event-B et l'outil RODIN Jean-Raymond Abrial ira plus loin dans ce principe de démarche progressive en y associant la notion de preuve de correction.

Plus modestement dans nos cours de bases de données et systèmes d'information dans les années 1980 nous avons choisi de proposer une démarche de conception spécifique centrée sur Z0 basé sur quelques principes simples pour construire progressivement des schémas « normalisés » qui dans une étape ultime sont transformés en schémas relationnels en 3FN. Par exemple nous distinguons les ensembles « centraux » comme PERSONNE, PRODUIT et VOITURE des ensembles « primitifs » comme ENTIER, CHAINE, COULEUR, etc. Nous construisons un sous-schéma normalisé pour chaque ensemble central qui contient au moins une association identifiante ((1,1), (0,1)) avec un ensemble primitif qui peut être défini par une expression de calcul sur des ensembles primitifs : par exemple l'immatriculation d'une voiture par un triplet. Pour chaque ensemble central une check liste de questions aide à ne pas oublier d'associations avec des ensembles primitifs en s'intéressant aux divers rôles possibles : identifiant, quantitatif, qualitatif, catégoriel, spatial, temporel. Toutes les contraintes sont à exprimer sur les sous-schémas et le schéma global qu'elles soient d'unicité, identification, totalité, exclusion ou inclusion. Au moment de l'intégration des sous-schémas dans un schéma global il faut porter une attention particulière aux noms des ensembles et fonctions de dépendance pour éviter les problèmes de polysémie, homonymie et synonymie, appliquer des règles de généralisation (∪ d'ensembles) et de spécialisation ({… | …}, partitions) pour permettre des héritages structurels et diminuer le nombre d'associations. Les patterns de conception du génie logiciel sont aussi utilisables.

Dans l'étape finale la construction des schémas relationnels pour une base de données est basée sur le principe qu'une association Z peut être traduite systématiquement par une relation de Codd comme dans la figure 12.

L'expression Z0 avec les ensembles (E_1, E_2, … E_n), les fonctions F et G et l'association

$$E_1 \times E_2 \times \ldots \times E_k \xrightarrow[G]{F} E_{k+1} \times E_{k+2} \times \ldots \times E_n$$

est traduite par la relation FG :

$FG\ (C_1 : E_1, \ldots C_k : E_k, C_{k+1} : E_{k+1}, \ldots C_n : E_n)$

où C_i sont des constituants, E_i des domaines

et $(e_1, e_2, \ldots e_n) \in FG \equiv (e_1, e_2, \ldots e_k) \in G\ (e_{k+1}, e_{k+2}, \ldots e_n)$

$\equiv (e_{k+1}, e_{k+2}, \ldots e_n) \in F\ (e_1, e_2, \ldots e_k)$

Figure 12 – Traduction relationnelle d'une association Z0.

Les associations Z0 identifiantes donneront les clés des relations. Sur ces principes un outil peut être construit pour un processus de conception de schémas relationnels normalisés à partir d'un schéma Z0.

Utilisation de Z0 dans un processus alternatif de conception d'une BD relationnelle en 3FN

Pour obtenir un schéma relationnel vérifiant une propriété P (en 3e forme normale) une approche traditionnelle consiste à partir d'un schéma relationnel R1 en 1re forme normale, à atteindre la propriété P par des transformations successives jusqu'au schéma R en 3e forme normale. La figure 13 schématise l'approche alternative que nous avons proposée basée sur un schéma conceptuel S en Z0 et sur une propriété P' analogue à P, car la notion de normalité relationnelle n'est pas définie en Z. Après analyse du schéma S, des transformations successives sont réalisées pour atteindre un schéma Sp respectant la propriété P'.Nous avons implanté ce processus alternatif de normalisation en 1985 dans un système expert prototype réalisé en s'appuyant sur un moteur mixte PROLOG-OPS5 [Giraudin et coll. 1985a].

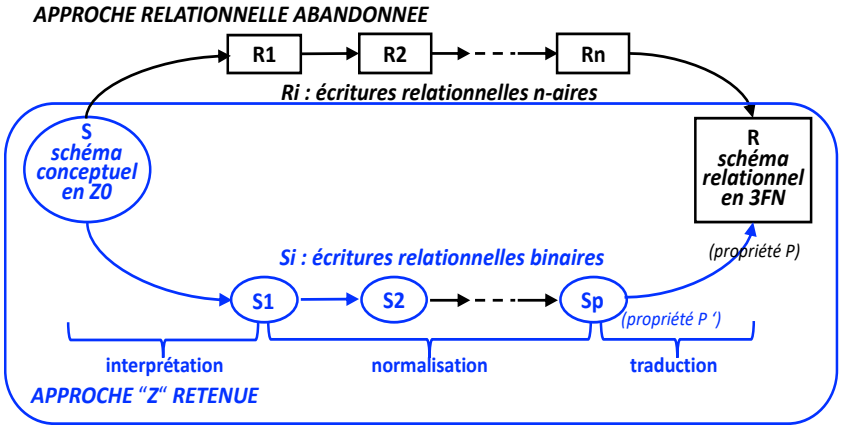


Figure 13 - Processus alternatif de normalisation.

Le choix d'une expression initiale d'un schéma conceptuel de données en Z0 est le résultat de plusieurs constats. Dans le modèle relationnel, la notion de dépendance fonctionnelle n'est pas suffisante à elle seule pour représenter formellement des faits. Une dépendance fonctionnelle peut représenter un fait, une contrainte, les deux, ou ni l'un ni l'autre. Si d'autres dépendances formelles sont proposées dans le modèle relationnel (dépendances hiérarchiques, multivaluées, produits, etc.), leur fondement sur une seule relation universelle à décomposer en sous-relations ne permet pas d'exprimer toute la sémantique d'une Base de Données et leur usage est difficile pour les concepteurs. Les concepteurs ne démarrent pas la conception d'un schéma de Base de Données sur une relation unique (une table relationnelle unique contenant toutes les données utiles).

Assistance à la conception de Systèmes d'Information basée sur Z

Nous avons conduit de nombreux autres projets avec Z pour une Assistance à la conception de SI. C'est le cas à la fin des années 1980 dans le cadre du projet Système d'Information Hospitalier Intégré à l'hôpital de Grenoble et d'une coopération avec François Bodart de l'université de Namur, avec l'outil IDA (Interactive Design Approach) issu des travaux de Daniel Teichroew sur le projet ISDOS ((Information System Design and Optimization System). Nous avons développé un atelier de Génie Logiciel combinant des modèles et des langages au sein d'outils de type système expert pour guider un analyste dans l'usage d'un logiciel d'aide à la spécification et à la conception de SI. Les modèles IDA de description de SI et l'ensemble de règles initialement écrites en langue naturelle à vérifier sur ces descriptions sont réécrits sous forme d'énoncés Z [Alvares et coll. 1989].

Les années 1980 voient un foisonnement de propositions au niveau des langages (approches fonctionnelles, logiques, logiques descriptives, objets, etc.) et des moteurs d'inférence (chaînage avant ou chaînage arrière des règles, Base de Connaissances monotone ou non, choix d'états ou résolution de conflits, etc.). Chaque langage ou approche apporte ses notations spécifiques, les notations graphiques se multiplient pour les boîtes et les flèches et n'arrivent à ne parler qu'aux spécialistes. Nos expériences d'usage combiné du langage Z0 et de solutions de type systèmes experts nous ont convaincu de la capacité sémantique du langage Z0 pour structurer formellement les données et que c'était un bon candidat pour servir de langage pivot au sein d'outils de conception. Ces notations mathématiques classiques connues de tous les scientifiques permettent aux concepteurs de se concentrer sur l'acquisition des connaissances et la validation d'une solution conceptuelle. Ces éléments nous ont incité à publier l'article *« Pour une réhabilitation du modèle relationnel binaire »* dans la revue de Génie Logiciel BIGRE-GLOBULE en 1985 [Giraudin et coll. 1985b].

À la fin des années 1990, nous nous sommes intéressés à de nouvelles formes de couplage-transformation entre langages ou modèles au sein d'un atelier de conception de SI. Dans le but de mieux concilier et intégrer différents points de vue (syntaxique, sémantique, statique, dynamique, fonctionnel, etc.) pour construire des SI plus cohérents et plus sûrs, deux techniques générales de modélisation sont utilisées : la multi-modélisation et la méta-modélisation. Nous avons développé un prototype d'atelier de modélisation, multi-modélisation et méta-modélisation pour faciliter l'usage combiné de formalismes (informels, semi-formels, graphiques, formels) et des transformations inter-formalismes [Freire et coll. 1998]. La multi-modélisation est utilisée à deux fins : exprimer la même chose selon des formalismes différents ou exprimer des vues différentes de la même chose dans un même type de formalisme. La sémantique précise des langages Z et Object-Z les rend aptes à des vérifications de cohérence des spécifications d'un système.

4- Pédagogie par l'exemple associant Z0, Z1 et SOCRATE

4.1- Diffusion pédagogique de Z0, Z1 et SOCRATE

Jean-Raymond Abrial assure à Grenoble dès 1972 des enseignements en maîtrise d'informatique et en DEA sur les structures de données et les programmes, les espaces de données virtuels et physiques, les multiples liaisons entre les espaces d'une donnée nécessaires à leur gestion dynamique en mémoire principale et sur les mémoires secondaires : adresses, tables d'occupation, catalogues des noms, tables des symboles, etc. L'environnement pédagogique grenoblois a bénéficié en primeur de manuels utilisateurs pour la version prototype de SOCRATE sur IBM 360/67 avec le système CP/CMS en 1975 et pour Z. Ces manuels ont été déterminants pour organiser des enseignements SOCRATE et Z dans plusieurs

formations universitaires grenobloises (IUT, MIAGE, MINF, ENSIMAG, etc.) en bases de données, programmation et spécifications de systèmes d'information.

Le livre « *Méthodes de programmation* » en 1978 de Bertrand Meyer et Claude Baudoin fait connaître plus largement Z. Il contient une quinzaine de pages sur Z. Ces pages donnent tous les éléments utiles pour une bonne compréhension et utilisation de Z0 et Z1. La discussion en fin de section rappelle l'importance de bien établir la spécification statique en Z0 de l'application avant de passer à l'étape algorithmique en Z1, donc au programme.

En 1982, le livre (rouge) de Claude Delobel et Michel Adiba « *Bases de données et systèmes relationnels* » contient lui aussi une présentation de Z0 pour modéliser des schémas conceptuels et Z1 comme LMD (langage de manipulation de données). Les langages de définition et de manipulation SOCRATE sont présents aussi dans cet ouvrage ainsi que l'introduction au langage QBE (Query By Example) développé par IBM combinant un langage prédicatif à variables domaines et une interface basée sur les tables du modèle relationnel. L'interrogation commence par afficher des tables vides (squelettes de relations) et d'utiliser un langage simple pour énoncer des opérations de l'algèbre relationnelle (sélection, projection, jointure, etc.) avec des formules et des raisonnements par analogie dans les cases des tables.

4.2- PPE – Pédagogie Par l'Exemple

Dès mon projet de DEA en 1973-1974 j'ai utilisé SOCRATE. Ensuite dans ma thèse de 3ème cycle en 1974-1977 je me suis inspiré de *data semantics* pour définir un modèle de transactions et un modèle conceptuel de données étendu pour des contrôles de gestion et des statistiques avec l'ajout par exemple des agrégats ensembles d'ensembles. Les fonctions d'accès définissant les associations binaires étaient renommées fonctions de dépendance. Pour la rentrée de septembre 1980, Claude Delobel m'a demandé de préparer des TD pour illustrer ses cours en Bases de Données et Systèmes d'Information assurés à la MIAGE de Grenoble. J'avais lu les pages sur SOCRATE, Z et QBE avant l'édition du livre qu'il finissait de rédiger avec Michel Adiba [Delobel et coll. 1982]. J'ai alors proposé en complément de quelques TD classiques la préparation d'une étude de cas basée sur une pédagogie par l'exemple. Cette étude de cas est composée de deux pochettes regroupant des documents.

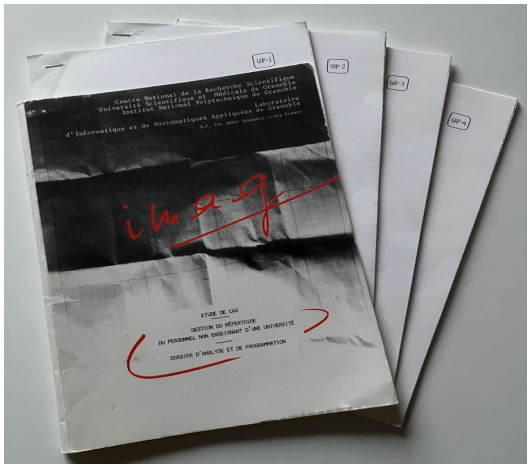


Figure 14 – Pochette « Étude de cas – Gestion du répertoire du personnel non enseignant d'une université – Dossier d'analyse et de programmation ».

La première pochette en photo figure 14 est intitulée « *Étude de cas – Gestion du répertoire du personnel non enseignant d'une université – Dossier d'analyse et de programmation* ». C'est une introduction aux dossiers qu'un analyste-programmeur doit écrire dans le cadre d'un projet professionnel. Sept chapitres composent cet ensemble de dossiers : 1) Description générale, 2) Analyse de l'existant, 3) Traitements envisagés, 4) Modèle conceptuel de données, 5)

Spécification des traitements, 6) Dictionnaire, 7) Base de données Socrate. Ces chapitres sont regroupés dans 4 documents préfigurant l'organisation du travail en TD : GRP-1 regroupe des 3 premiers chapitres, GRP-2 et GRP-3 contiennent respectivement les chapitres 4 et 5, GRP-4 pour les chapitres 6 et 7. Cette première pochette de près de 80 pages est largement inspirée d'un travail conduit à la fin des années 1970 par Dominique Portal au sein du service de gestion du personnel de l'Université Scientifique et Médicale de Grenoble.

La première séance de cette étude de cas en MIAGE 2e année consiste à distribuer cette première pochette, la commenter, répondre à des questions, réaliser de petits exercices avec les différents langages utilisés (Z et SOCRATE). Les étudiants disposent ainsi d'une application assez complète : analyse de l'existant, modèle conceptuel de données en Z0, spécification d'une dizaine de traitements sous forme de schémas transactionnels et procédures algorithmiques, Base de Données SOCRATE et programmes SOCRATE des traitements spécifiés.

À partir de ce dossier d'analyse-programmation type, les étudiants sont mis en situation pour traiter partiellement par analogie une application plus importante et complexe[8] présentée dans une seconde pochette « *Étude de cas – Gestion de la scolarité des universités grenobloises* ».

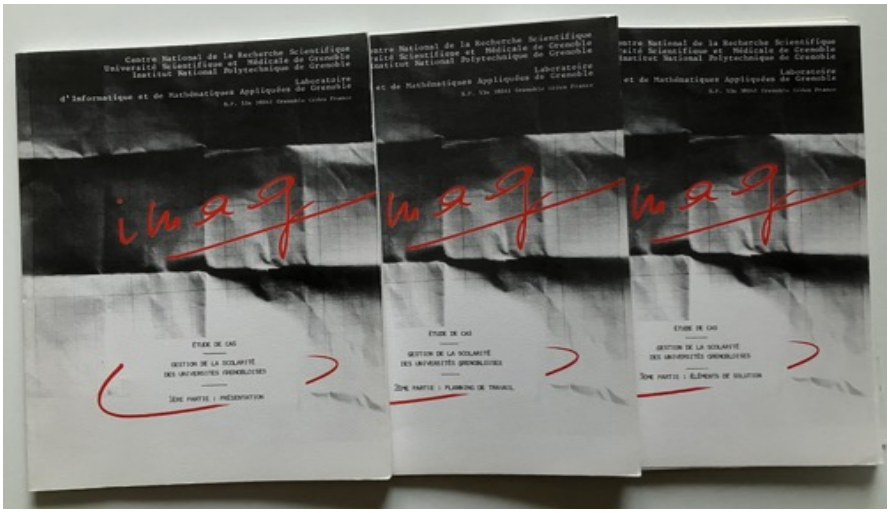


Figure 15 – Pochette « Étude de cas – Gestion de la scolarité des universités grenobloises ».

La figure 15 montre cette pochette organisée en 3 parties. Les 2 premières parties sont distribuées aux étudiants. La 1ère partie (Présentation) présente le contexte, les données manipulées et les traitements envisagés. La 2e partie (Planning de travail) présente une dizaine de tâches à réaliser en modélisation de données, spécification et réalisation de traitements. Le planning de travail est complété par des normes à utiliser pour la présentation des solutions : conventions d'écritures de dessins des graphes de données ou de transactions et des fiches standards de description des éléments conceptuels et de réalisation. A l'époque l'usage de fiches standards fortement structurées était très répandu dans les services informatiques pour composer des dossiers qui passaient d'un acteur à un autre (chef de projet, analyste, concepteur, programmeur) avec des allers-retours pour validation-correction.

Les documents de la 3e partie (Éléments de solution) sont destinés uniquement à l'enseignant qui encadre un groupe d'étudiants afin qu'il soit en mesure d'orienter les étudiants bloqués ou qui divergent, vers des solutions réalistes du type de celles mises en place au service de scolarité de l'USTMG (Université Scientifique, Technologique et Médicale de Grenoble). Le SGBD SOCRATE, développé à l'Université de Grenoble, était utilisé dans des applications de gestion

[8] Les dernières spécifications à réaliser sont liées à des problèmes non triviaux comme la gestion des cursus universitaires et des emplois du temps.

de cette université ce qui nous a permis d'utiliser de multiples documents et listings du service informatique pour construire cette étude de cas et des éléments de solutions. Après 2 ou 3 ans d'utilisation cette étude de cas a été remplacée par des projets avec la méthode Merise très fortement demandée par les entreprises et les sociétés de service.

5- Conclusion : pourquoi Socrate, *data semantics*, Z ?

La spécification et la réalisation du prototype SOCRATE peuvent être vues comme une commande conjointe du laboratoire IMAG et de l'hôpital de Grenoble à Jean-Raymond Abrial. Plus tard ne disait-il pas « *Si c'est important pour vous, vous trouverez un moyen. Sinon, vous trouverez une excuse* » ? Le besoin d'apporter plus de sémantique aux descriptions de structures de données se concrétise ensuite dans *data semantics* (1973) et la thèse de Rui Barbosa (1975).

Mais pourquoi Jean-Raymond Abrial a-t-il défini le nouveau langage Z en 1977-1978 et pourquoi baser ce langage sur la théorie des ensembles et des notations algorithmiques ? Une réponse simpliste serait « pourquoi pas ! ». Le chapitre 11 du livre de la Genèse nous donne une première réponse : « Dieu dit : *confondons leur langage pour qu'ils ne s'entendent plus les uns les autres».* D'autres langages sont proposés à l'époque à fortes bases mathématiques pour faire de la programmation fonctionnelle comme LISP créé en 1958, de la programmation logique comme PROLOG créé en 1972, ou encore le langage APL d'Iverson en 1966 avec son typage dynamique et ses notations mathématiques, etc. Chaque langage a sa spécificité, son domaine privilégié d'application ou un intérêt commercial. La réussite de Z tient au fait qu'il est typé, permet des structures de données et intègre des transformations successives vers de la programmation algorithmique. B franchit le pas suivant en associant un langage de spécification mathématique (comme Z), une modularité des descriptions, une méthode formelle de spécification et conception, et un outil d'assistance à la preuve.

Ce document n'a pas la prétention d'établir tous les apports de Jean-Raymond Abrial, ni d'être exhaustif et complètement fidèle à la syntaxe pour présenter les langages. Il doit être vu simplement comme le témoignage d'un enseignant-chercheur héritier de cours et travaux de recherche de Jean-Raymond Abrial mais aussi de Claude Delobel et François Peccoud, deux autres pionniers grenoblois dans les domaines des bases de données et des systèmes d'information. Dans ce document nous avons utilisé principalement des documents de Jean-Raymond Abrial et nos propres notes de cours et d'exercices des années 1970-1980 sans en modifier l'essence. Nous avons volontairement limité les références bibliographiques aux travaux grenoblois à l'exception du livre de Bertrand Meyer et Claude Baudoin.

Les contributions de Jean-Raymond Abrial à la communauté scientifique informatique sont bien plus importantes que celles présentées sommairement dans ce document. Le lecteur peut consulter avec intérêt sa conférence au Collège de France en 2015 intitulée « *Spécification, construction et vérification de programmes : le parcours d'une pensée scientifique sur une quarantaine d'années* »[9]. Il y décrit très bien sa quête de la spécification idéale et ses résultats tant théoriques que concrets.

Notes bibliographiques

Sous la direction de Michel Adiba et Jean-Pierre Giraudin, la maison d'édition de l'université de Grenoble a publié en 2024 un ouvrage collectif volumineux de 924 pages qui retrace 60 ans d'histoire de recherches grenobloises au sein de l'IMAG dans les domaines des données,

[9] https://www.college-de-france.fr/site/gerard-berry/seminar-2015-04-01-17h30.htm

informations et connaissances [Adiba et coll. 2024]. Dans la première partie de cet ouvrage nous présentons Jean-Raymond Abrial comme l'un des trois pionniers des recherches grenobloises dans ces domaines. La revue Specif, dans son numéro 46, publie en 2000 un témoignage de Louis Bolliet sur les premières années de l'IMAG [Bolliet 2000].

Les publications de l'équipe de Jean-Raymond Abrial sur le système SOCRATE sont nombreuses dans la décennie 1970 dont 6 thèses [Abrial et coll. 1970a, Abrial et coll. 1970b, Beaume 1970, Morin 1970, Vigliano 1970, Mazaré 1973, Barbosa 1975, Portal 1975, Stiers 1975]. L'ouvrage de Claude Delobel et Michel Adiba sur les bases de données et systèmes relationnels [Delobel et coll. 1982] décrivent les structures de données et le LDD SOCRATE. Le mémoire de master de Jean Ricodeau [Ricodeau 2016] développe largement l'histoire de SOCRATE de ses débuts à ses espoirs puis ses difficultés industrielles.

Les premières publications de Jean-Raymond Abrial sur des langages plus formels comme *data semantics* et Z datent aussi des années 1970 [Abrial 1973a, Abrial 1973b, Abrial 1974, Abrial 1977a-b-c]. Deux ouvrages à vocation pédagogique l'un en programmation [Meyer et coll. 1978], l'autre en bases de données [Delobel et coll. 1982] donnent une bonne connaissance générale de Z0 et Z1. L'équipe de recherche grenobloise en systèmes d'information a développé plusieurs projets avec Z, en particulier pour aider à la conception d'une base de données relationnelle [Giraudin et coll. 1985a, 1985b] ou pour expérimenter de nouvelles fonctionnalités dans des ateliers de génie logiciel [Alvares et coll. 1989, Freire et coll. 1998].

Références bibliographiques

[Abrial et coll. 1970a] Abrial J.-R., Bas J., Beaume G., Henneron G., Morin R., Vigliano G. (1970, août). Projet SOCRATE (1) – Spécifications générales. Communication Institut de Mathématiques Appliquées, Université de Grenoble.

[Abrial et coll. 1970b] Abrial J.-R., Beaume G., Henneron G., Morin R., Vigliano G., Valois J., Cohen S. (1970). Système de définition et interrogation de données : application au dossier médical. Symposium de Toulouse.

[Abrial 1972] Jean-Raymond Abrial (1972, janvier). Structure de données et de programmes (cours C4). Université de Grenoble.

[Abrial 1973a] Jean-Raymond Abrial (1973). Data semantics. Workshop, Alpe d'Huez.

[Abrial 1973b] Jean-Raymond Abrial (1973, août). Description sémantique des bases de données. Notes pour le cours donné au 7th workshop in the IAG Data Base Series à Bruxelles.

[Abrial 1974] Jean-Raymond Abrial (1974, avril). Data semantics. IFIP TC2 Working conference on DataBase Management, Cargèse, Corse, dir. Klimbie et Koffeman (p. 1-59), North-Holland. Article diffusé par le Laboratoire d'Informatique de l'Université Scientifique et Médicale de Grenoble.

[Abrial 1977a] Jean-Raymond Abrial (1977). Manuel du langage Z (Z/13). IMAG, Grenoble.

[Abrial 1977b] Jean-Raymond Abrial (1977, juin). Mécanismes de transformations du langage Z (Z/14). Notes internes Z/1 à Z/15 non publiées, EDF.

[Abrial 1977c] Jean-Raymond Abrial (1977, novembre). Utilisation du langage Z pour l'analyse d'une petite application de gestion (Z/15). Notes internes Z/1 à Z/15 non publiées, EDF.

[Adiba et coll. 2024] Michel Adiba, Jean-Pierre Giraudin (2024, 7 novembre). Du Big Data à l'IA - 60 ans d'expérience en traitement des données, des informations et des connaissances à Grenoble. UGA Editions, Prométhée, Grenoble.

[Alvares et coll. 1989] Alvares Luis, Giraudin Jean-Pierre, Chabre-Peccoud Monique (1989, mars). Des bases pour piloter la modélisation des systèmes d'information par une approche IA. CIL 89 (p. 396-416). Barcelone.

[Barbosa 1975] Rui Barbosa (1975, 11 octobre). Contribution à l'étude sémantique des bases de données, application au système SOCRATE. Thèse de docteur-ingénieur en informatique de l'Université Scientifique et Médicale, Grenoble.

[Beaume 1970] Georges Beaume (1970, 22 décembre). Projet SOCRATE (2.2) – Gestion des Mémoires. Doctorat de mathématiques appliquées de l'Université de Grenoble.

[Bolliet 2000] Louis Bolliet (2000, novembre). Témoignage sur les premières années de l'IMAG. Dans revue Specif n° 46, p. 49-52.

[Delobel et coll. 1982] Delobel C., Adiba M. (1982). Bases de données et systèmes relationnels (pages Z0-Z1 : 18-25, 117-131). Dunod Informatique.

[Freire et coll. 1998] J.C. Freire Junior, J-P. Giraudin, A. Front, M. Chabre-Peccoud (1998, septembre). A CASE tool for the Modeling of Methods and Information System. OOIS 98, International Conference on Object Oriented Information System (p. 387-404). Paris. Edt Springer Verlag.

[Giraudin et coll. 1985a] Giraudin J-P., Delobel C., Dardailler P. (1985, mars). Éléments de construction d'un système expert pour la modélisation progressive d'une Base de Données. Acte des journées sur les Bases de Données Avancées, Saint-Pierre-de-Chartreuse.

[Giraudin et coll. 1985b] Giraudin J-P., Chabre-Peccoud M. (1985, octobre). Pour une réhabilitation du modèle relationnel binaire. Acte des journées « Nouveaux langages pour le Génie Logiciel ». Revue BIGRE-GLOBULE n° 45.

[Mazaré 1973] Guy Mazaré (1973, 18 mai). Projet SOCRATE (8) – Structure d'un système d'exploitation adapté à la base de données. Thèse de docteur-ingénieur en informatique de l'Université Scientifique et Médicale de Grenoble.

[Meyer et coll. 1978] Bertrand Meyer, Claude Baudoin (1978). Méthodes de programmation (pages Z0-Z1 : 526-542). Eyrolles.

[Morin 1970] Robert Morin (1970, 22 décembre). Projet SOCRATE (2.3) – Interprétation et Compilation. Doctorat de mathématiques appliquées de l'Université de Grenoble.

[Portal 1975] Dominique Portal (1975, 20 mars). Conception d'un système automatisé de gestion de la scolarité de l'enseignement supérieur. Thèse de docteur-ingénieur en informatique de l'Université Scientifique et Médicale de Grenoble.

[Ricodeau 2016] Jean Ricodeau (2016, juin). Le cycle de vie de SOCRATE, logiciel informatique de bases de données, de 1963 à 1990 – Parcours professionnels et innovations, à Grenoble territoire de coopérations Université-Entreprises. Mémoire de Master 1 Sciences humaines et sociales, Université Grenoble Alpes.

[Stiers 1975] Andrée Stiers (1975). Manuel d'utilisation de SOCRATE. Institut de Mathématiques Appliquées, Grenoble.

[Vigliano 1970] Georges Vigliano (1970, 22 décembre). Projet SOCRATE (2.1) – Langage de requête. Doctorat de mathématiques appliquées de l'Université de Grenoble.