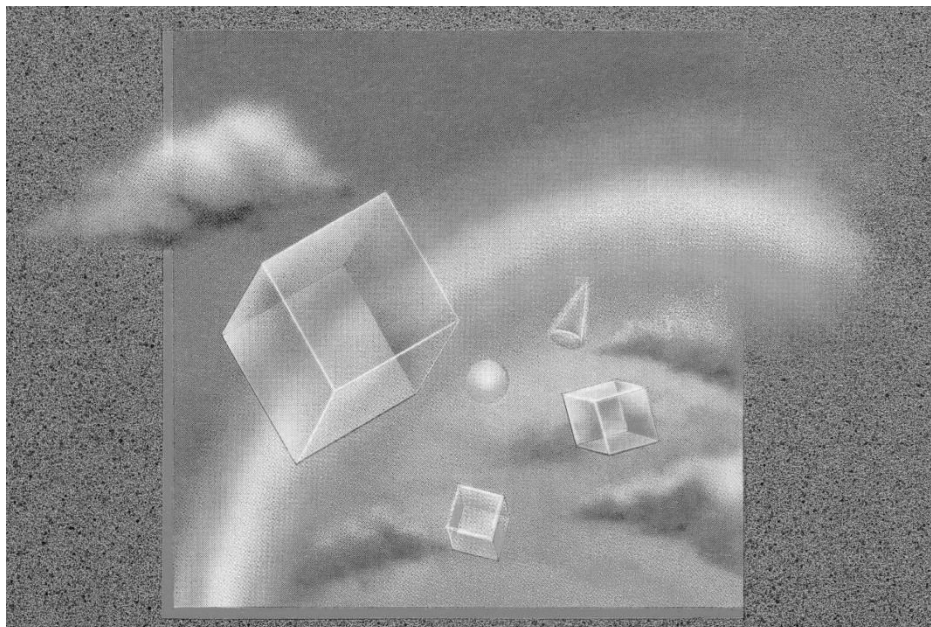


Génie Logiciel Assisté par Ordinateur



Hiver 2017

Réal Gendron

Table des matières

GLAO 4 pages

Choisir ses outils de développement 7 pages

Les AGL (Ateliers de génie logiciel) 15 pages

GLAO = Génie Logiciel Assisté par Ordinateur
CASE = Computer Aided Software Engineering

Définition

- Outils qui réduisent ou éliminent substantiellement les problèmes inhérents à la conception et au développement des moyens et gros projets en **générant automatiquement la plupart du logiciel selon les spécifications de l'architecte du logiciel**
- Outils qui produisent un levier à n'importe quel point dans le cycle de développement du logiciel, plus particulièrement dans les phases d'analyse des exigences et conception et qui potentiellement génèrent le code automatiquement à partir des spécifications
- Approche disciplinée et structurée de développement de systèmes qui **permet au concepteur de se concentrer sur l'architecture du système plutôt que son implantation.**

Les outils GLAO proviennent des méthodologies structurées de développement de systèmes qui datent des années 1960 et 1970.

Taxonomie des outils GLAO

1- L3G (langages de 3^e génération)

- Langage de programmation tel que **C, COBOL, BASIC, PASCAL**
- Outil simple extrêmement efficace quant à la qualité du produit fini
- Ne couvre **qu'une infime partie du cycle de développement** et exige une expertise de programmeur

2- Outils micro

- **Chiffrier électronique (Excel, etc.), SGBD (Access, etc.) et autres outils de développement (Visual Studio, etc.)**
- S'apparentent aux L4G (langages de 4^e génération)
- **Peut gérer des applications en réseau et des bases de données**

3- L4G (langages de 4^e génération) et SGBDR (système de gestion de bases de données relationnelles)

- Principalement pour la mini et la grande informatique
- Exemples : **Microsoft SQL Server, Oracle, Informix, Procol, etc.**
- Couvrent un grand spectre du cycle de développement d'une application et

peuvent être utilisés comme seuls outils de développement

- Souvent, SGBDR existant sur plusieurs plateformes technologiques (Unix, Windows, etc.), multi-usagers, fonctionnant en réseau, offrant un dictionnaire de données et un langage de requête (SQL habituellement), comportant un module de design d'écran et de rapport, pouvant générer des applications avec des interfaces usagers graphiques et s'interfacer avec d'autres outils de développement

Les outils précédents ne couvrent qu'une partie du cycle de développement. Pour une couverture plus complète il faut utiliser un **GLAO intégré** ou un *atelier de génie logiciel* (AGL) constitué à partir d'outils spécialisés dans une seule partie du cycle.

4- GLAO intégré

- Couvre une grande partie du cycle
- Pas vraiment complet

5- GLAO supérieur (Upper CASE)

- Porte sur les étapes de planification, conception, architecture et analyse (Exemple Silverrun, Open ModelSphere)

6- GLAO inférieur (Lower CASE)

- Porte principalement sur la construction (programmation)

7- GLAO en modules

- Se concentre sur une fonction plus pointue
- Peut s'interfacer avec d'autres GLAOs

Fonctions des outils de développement

A) Gestion de projets

- Surtout sur micro-ordinateur (exemple Microsoft Project)

B) Modélisation d'entreprises

- Pour rédiger un plan directeur (mission, structure, etc.)
- Pour produire un organigramme d'entreprise
- Traitement de textes

C) Architecture des systèmes

- Outil souvent graphique (System chart, etc.)

D) Aide à l'analyse

E) Modélisation des données (DER ou MCD)

- Choix entre une multitude de méthodes (Chen, Yourdon, Merise, etc.)
- Modélise la structure des données, les flux de données, les bases de données et les vues (VIEWS)

F) Modélisation des traitements

- Choix entre une multitude de méthodes (Gane & Sarson, Yourdon, etc.)
- Séquence et définition des traitements

G) Modélisation UML (pour l'approche orientée objet)

- Produit divers diagrammes

H) Design d'écrans et de rapports

- Plus ou moins ergonomique (SCREEN PAINTER)

I) Design automatique des bases de données

- Langage de définition de données (DDL) (par exemple SQL)

J) Dictionnaire des données

- Définition et attributs des variables
- Cœur de l'application et du système de développement

K) Référentiel

- Plus complet que le dictionnaire de données
- Contient des informations sur tous les « objets » (modèles, fichiers, bases de données, traitements, etc.) nécessaires à la conception, au développement, à la modification et à la documentation
- Cœur de l'application et du système de développement

L) Génération de documentation

- À partir du dictionnaire ou du référentiel

M) Vérification de consistance

- À partir du dictionnaire ou du référentiel
- Vérifie les définitions et les traitements entre eux

N) Maquettage/Prototypage

- Construction des écrans (maquettage) et simulation de certains traitements (prototypage)
- Pour que l'utilisateur final puisse tester et donner son impression afin de revenir à la conception pour réajuster le tir

O) Génération de code

- Selon la sophistication de l'outil, cette fabrication sera plus ou moins automatique
- Recours aux L3G (langages de 3^e génération) et aux langages évolués
- Certaines sections délicates sont programmées « à la main »

P) Génération d'interfaces usagers graphiques (IUG)

Q) Tests

- Simple débogage (trace, compilation, point d'arrêt [BREAKPOINT], vidage mémoire [MEMORY DUMP], exécution inverse, etc.)
- Génération de masses de données combinées (CROSS REFERENCE)

R) Réingénierie

- Transforme un vieux système en application moderne et structurée
- Constitution de bibliothèques
- Analyse et conception orientée objet (OOAD = Object-oriented analysis and design)

Avantages des outils GLAO

- 1- Réduit souvent le temps de développement (productivité)
- 2- Meilleure qualité du produit
- 3- Connaissance plus complète des spécifications
- 4- Automatisation des tâches des développeurs
- 5- Haut degré de standardisation
- 6- Documentation exacte et consistante
- 7- Respecte mieux les exigences des usagers
- 8- Maintenance plus rapide

CHOISIR SES OUTILS DE DÉVELOPPEMENT

PAR GIL TOCCO (Revue INFORMATIQUE & BUREAUTIQUE décembre 1992)

Que ce soit pour bâtir une application individuelle ou un énorme système transactionnel, il existe des outils de développement. Que peuvent-ils faire pour vous et comment choisir. Ce dossier vous aidera à répondre à ces questions.

Quelle que soit la taille d'une application à développer, ceux qui en ont la charge devront passer à travers un certain nombre d'étapes. Ces étapes ont été plus formalisées depuis l'apparition du concept de génie logiciel, mais elles ont en fait toujours existé, ce sont les suivantes:

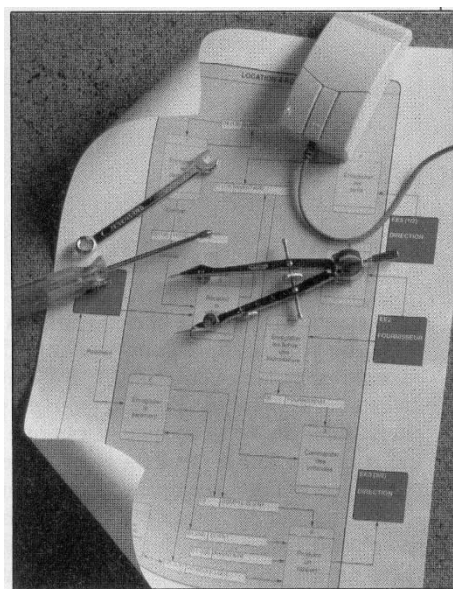
- la planification;
- la conception et l'analyse;
- la programmation;
- l'implantation;
- les tests;
- l'entretien.

Si ces étapes sont scrupuleusement suivies lors du développement de grands projets, il n'en est pas toujours de même pour les projets de moindre importance. Ainsi, on oubliera souvent les premières étapes (planification, conception) et la dernière (entretien).




De nombreux outils informatisés existent à tous les niveaux. S'ils sont trop coûteux ou trop complexes par rapport à la taille du projet, les développeurs devraient quand même passer à travers ces étapes en utilisant des méthodes manuelles.

Il est en effet impensable de développer un système, aussi petit soit-il, si on n'a pas analysé son impact sur l'entreprise, ou au moins sur l'unité administrative dans laquelle il sera utilisé. Si on ne le planifie pas, on risque fort de générer une multitude de coûts cachés mais néanmoins réels. Si l'analyse n'est pas scrupuleuse, on héritera d'une application qui ne fonctionne pas ou ne correspond pas aux besoins.

Le tableau I ci-contre vous indique les fonctions que chaque type d'outil vous permet d'automatiser. S'il ne les automatise pas toutes, les autres devront être traitées manuellement.



	L3G	Outils de micro	SGBD	L4G	GLAO supérieur	GLAO inférieur	GLAO intégré	GLAO en modules
Gestion de projet								
Modélisation d'entreprise								
Architecture des systèmes								
Aide à l'analyse								
Modélisation des données								
Modélisation des traitements								
Design d'écran								
Design automatique de SGBD								
Dictionnaire								
Référentiel								
Génération de documentation								
Vérification de consistance								
Maquettage/Prototypage								
Programmation en L3G								
Programmation en langage évolué								
Génération de code								
Tests								
Génération des IUG								
Réingénierie								

 Fonction toujours incluse dans le produit
  Quelquefois incluse
  N'est pas incluse

LES OUTILS SIMPLES

Commençons par les outils simples. Les langages de programmation (ou de 3^e génération) sont extrêmement efficaces quant à la qualité du produit fini. Ainsi, la plupart des progiciels populaires sont développés en langage C. Borland, Microsoft ou Lotus arrivent ainsi à contrôler de très près la performance de leurs produits. Plusieurs programmeurs-maison utilisent le Basic, le Cobol ou le Pascal pour de petites applications.

Le défaut de ces langages, c'est qu'ils ne couvrent qu'une toute petite partie du cycle de développement et qu'ils nécessitent une expertise que l'on ne trouve pas partout; il faut être programmeur.

On a donc de plus en plus tendance, en entreprise et même dans certaines boîtes de logiciels, à utiliser des produits un peu plus évolués pour développer des applications sur micro-ordinateurs.

Les tableurs Lotus 1-2-3, Quattro Pro [Borland] et autres peuvent être considérés comme des outils de développement pour certaines applications financières. Des systèmes comptables complets, pour petites entreprises, ont été développés avec ces produits.

Les SGBD tels que dBASE, Paradox [Borland] ou FoxPro [Microsoft] et d'autres du même niveau sont de bons outils pour développer des applications centrées sur des bases de

données, mais d'envergure limitée. En fait, ces produits comprennent tous des langages qui s'apparentent aux L4G. Dans certains cas, ils ont évolué au point de devenir des langages très complets, comme dans le cas de Clipper [Nantucket] ou de dBFast [Computer Associates] qui permettent de mieux tirer parti des SGBD et de compiler les applications développées.

Un bon SGBD peut créer des applications qui fonctionneront en réseau avec tout ce que cela implique: système de verrouillage des fichiers ou des enregistrements, sécurité, etc. Cependant, le développement lui-même ne peut raisonnablement s'effectuer que par une seule personne, ces produits étant mal adaptés au travail de groupe.

D'autres produits de développement pour micros commencent à apparaître. C'est le cas de VisualBasic [Microsoft] par exemple. En plus d'un langage Basic, il comprend toute une panoplie de caractéristiques : compilation, construction d'interfaces graphiques, construction assistée de bases de données, système de débogage, interface avec le langage C, etc.

Dans cette catégorie d'outils, on trouve aussi PowerBuilder [Powersoft Corp.] et ToolBook [Asymetric Corp.]. Nous avons regroupé tous les produits de micro-informatique (sauf les L3G) dans la colonne *Outils de micro* de nos deux tableaux.

SGBDR ET L4G

Pour s'attaquer à des applications plus solides, il faudra faire appel à d'autres outils, soit les L4G et les SGBD relationnels provenant de la mini ou de la grande informatique. Il est souvent difficile de faire la différence entre ces deux types d'outils. En effet, tous les SGBDR utilisent un L4G, même s'il est souvent offert en option et la plupart des L4G comprennent un SGBDR.

Ces produits, Oracle, Sybase, Informix, Ingres, Procol, etc., peuvent soit être utilisés comme seuls outils de développement, soit s'intégrer dans des processus plus complets (génie logiciel). Grâce à plusieurs caractéristiques, ils couvriront un plus grand spectre du cycle de développement d'une application. À l'achat d'un SGBDR ou d'un L4G, il faudra s'intéresser à ces caractéristiques :

- Le produit existe en versions fonctionnant sous plusieurs systèmes d'exploitation, par exemple Windows, Unix, VAX/VMS et OS/400. Même si vous n'envisagez de développer que sous un seul environnement, les choses peuvent changer dans un, deux ou cinq ans.
- Il est multi-usagers. Il peut permettre de fabriquer une application multi-usagers, mais il faudra aussi qu'il puisse fonctionner lui-même en multi-usagers si l'application est développée par plusieurs personnes à la fois.
- Il fonctionne en réseau et construit des applications fonctionnant en réseau.
- Il offre un dictionnaire des données ou de système qui sera fort utile lors de la documentation et de l'entretien.

- Il offre un langage de requête (préféablement SQL) qui permettra de construire très rapidement des demandes ponctuelles.
- Il comprend un module de design d'écran.
- Il permet de générer des applications avec interfaces usagers graphiques.
- Il peut s'interfacer avec d'autres outils de développement.

Vous pourrez constater, en regardant le tableau I, que les caractéristiques peuvent beaucoup varier d'un produit à l'autre. C'est à vous d'étudier en détails les caractéristiques de chacun en y ajoutant d'autres critères de sélection. Par exemple le prix, la performance ou encore la simplicité. Un produit comme Procol, très simple d'utilisation et peu coûteux, permet de bâtir des systèmes complexes. Cependant, il n'a pas encore d'interfaces avec des modules de génie logiciel. Il vous faudra donc peser le pour et le contre.

LE CYCLE COMPLET

Encore une fois, les outils précédemment cités ne recouvrent qu'une partie du cycle de développement. Pour le recouvrir le plus complètement possible de façon automatisée, il existe deux solutions.

La première, c'est de se procurer un système complet de développement GLAO, soit un GLAO intégré (ICASE pour Integrated Computer Assisted Software Engineering en anglais). Il en existe peu et les plus connus sont PacBase [SIRC], IEF [Texas Instruments], Foundation [Andersen Conseil] et IEW/ADW [KnowledgeWare].

La seconde solution c'est de se monter un atelier de génie logiciel complet à partir de plusieurs produits spécialisés dans une seule partie de ce cycle. Dans ce cas, les produits ne manquent pas, en fait, ils pullulent.

Certains offrent des produits se concentrant sur les étapes de planification, conception, architecture et analyse. C'est ce qu'on appelle en anglais le "Upper CASE" et que nous avons appelé *GLAO supérieur* dans nos deux tableaux. Un produit québécois comme Silverrun [CSA Recherche] se retrouve dans ce créneau.

D'autres produits se concentrent plus sur la construction proprement dite; ce sont les "Lower CASE" que nous avons nommés *GLAO inférieur*.

Enfin, on peut trouver une multitude d'outils se concentrant sur une fonction encore plus pointue et qui peuvent s'interfacer avec d'autres produits de génie logiciel. Dans nos tableaux, nous les avons appelés *GLAO en modules*.

Notons que des produits "complets" comme AD/Cycle [IBM] ou Cohesion [Digital Equipment] ne sont pas si complets que ça. D'une part, ils sont en cours de développement (on attend toujours le fameux référentiel "Repository" d'IBM) et, d'autre part, ils

sont spécifiquement conçus pour que des tierces parties viennent y greffer leurs modules ou remplacer les modules déjà existants en respectant un certain nombre de normes.

Même les autres produits complets jouent un peu le même jeu puisque la plupart d'entre eux tentent de s'ouvrir au maximum sur le reste du monde. Par exemple, dans les modules d'analyse, ils laissent le choix entre plusieurs méthodes (Yourdon, Gane & Sarson, Chen, Merise, etc.). Ils permettent l'interfaçage avec un maximum de SGBD relationnels, les plus vendus évidemment.

On trouve de plus en plus d'entreprises qui préfèrent bâtir un atelier de génie logiciel complet à partir de multiples morceaux, à la manière d'un gigantesque puzzle plutôt que d'acquérir un système intégré qui ne correspond pas exactement à leurs façons de faire, à leurs besoins ou à leurs ressources.

FONCTIONS ET OUTILS

Le tableau I vous donne donc la liste des principales fonctions des outils de développement. Elles ne correspondent pas exactement aux différentes phases du cycle de vie d'un système. Il n'existe pas d'outils d'entretien proprement dits, par exemple. Cependant, nous avons tenté de les mettre dans un ordre qui correspond le mieux possible à ces phases.

Le développement d'une application étant un projet en soi, on peut avoir besoin d'un logiciel de *gestion de projet*. Ce dernier est bien souvent un produit de micro-informatique de portée générale.

Bien souvent, les outils de planification fonctionnent aussi sur des micros, même en GLAO intégré. En fait, il existe rarement de liens directs et automatisés entre ces outils et les suivants.

Par *modélisation d'entreprise*, nous entendons tous les modules d'aide qui permettent de rédiger un plan directeur. Il s'agit bien souvent de produits générant des diagrammes illustrant la structure de l'entreprise du point de vue des unités administratives, des hiérarchies, des activités. La partie la plus importante du travail provient directement des cerveaux des gestionnaires qui doivent réfléchir sur la mission de l'entreprise, sa structure, la relation entre les applications à développer et ces dernières caractéristiques. D'ailleurs, le simple traitement de texte est aussi un outil fort utile durant cette étape.

De la structure dynamique de l'entreprise, on passe à la structure de l'information. Une première étape consiste à bien comprendre qui utilise quel grand ensemble d'information (bases de données existantes ou à créer).

« À cette fin, explique Louis Lorrain, président de Palantir Conseil à I&B, on utilise des matrices comportant une série d'entrées par ensemble d'information d'une part et une autre série par ensemble d'utilisateurs. Le système tente ensuite de diagonaliser la

matrice. » À la fin de l'opération, on aura une idée bien plus claire des regroupements de données ou d'utilisateurs les plus adéquats.

L'étape d'après consiste à bâtir *l'architecture du système* (ou des systèmes) et à commencer *l'analyse* proprement dite. Ici encore, ce sont des outils graphiques qui aideront les concepteurs à formaliser cette architecture.

On passe ensuite à la *modélisation des données* et à la *modélisation des traitements*. Les outils évolués vous donneront le choix entre une multitude de méthodes (qu'on appelle couramment les méthodologies) et de représentations de diagrammes structurés.

La structure des données, les flux de données, la séquence des traitements étant définis, on peut passer à la fabrication en tant que telle. Selon la sophistication de l'outil, cette fabrication sera plus ou moins automatique. Cependant, même avec les outils les plus complets, il arrive souvent que certaines sections délicates soient programmées "à la main".

Certains sont suffisamment avancés pour effectuer automatiquement le *design des bases de données* (en DDL, le langage de définition bases de données SQL, par exemple). L'assistance au *design d'écran* est plus courante, bien que plus ou moins ergonomique.

Nous avons placé les *dictionnaires* et *référentiels* au centre de notre tableau car il s'agit du coeur à la fois de l'application et du système de développement.

Même dans des outils très simples, on peut trouver un dictionnaire des données. Il contient la définition et les attributs de chacune des variables, ce qui est fort utile durant tout le cycle de vie de l'application. En génie logiciel, on utilise un référentiel beaucoup plus complet. Il contient tous les modèles utilisés lors du développement, les spécifications des fichiers du système, de chaque base de données, de chaque variable, de chaque traitement, de chaque modification. En langage moderne, on dit que le référentiel contient des "objets", tous les objets nécessaires à la conception, au développement, à la modification, à la *documentation*, bref à la vie de ou des applications.

C'est lui aussi qui permettra de faire des *vérifications de consistance* entre les diverses définitions et les divers traitements. Avant d'en arriver à une version finale du système, on passe de plus en plus par une phase de *maquettage* et de *prototypage* qui consistent à construire les écrans (maquettage) et simuler certains traitements (prototypage) de façon à ce que l'utilisateur final puisse tester et donner son impression, ce qui permettra éventuellement de revenir à l'étape de conception pour rectifier le tir.

La *génération de code* sera soit automatisée soit assistée de *langages évolués* ou de *langages de programmation*.

De plus en plus, des outils ou modules permettent de *générer les interfaces usagers graphiques* (IUG).

Les outils de *tests* peuvent aller du simple débogage à la génération de masse de données combinées.

« Dans les très gros systèmes transactionnels, explique Louis Lorrain, un gros mainframe peut être utilisé pour générer de forts volumes de transactions alimentant une autre machine. »

Enfin, la *réingénierie*, si elle ne permet pas encore de transformer un vieux système en application moderne et structurée, fournit une panoplie d'outils qui facilitent cette tâche.

LEQUEL CHOISIR?

Nous avons voulu construire un tableau simple répondant à la question: « Quel outil choisir pour développer chaque type d'application? » Le problème, c'est qu'il existe des milliers de types d'applications.

Avec l'aide précieuse de notre ami Gérard Blanc, président de GBA-groupe conseil, nous avons réussi à construire le tableau II qui donne une partie de la réponse.

Quelques précisions s'imposent. Par *système transactionnel*, nous entendons tout système traitant les données provenant d'un fort volume de transactions. Toutes les autres applications sont *non transactionnelles*.

Les *transactions simples* demandent peu de traitement: enregistrement des permis de conduire (*centralisé*), magasins d'alimentation (*réparti*).

Les *transactions complexes* demandent soit d'importants calculs, soit beaucoup de manipulation d'information, soit les deux : compagnie aérienne (*centralisé*), banque (*réparti*).

La différence entre le *non-transactionnel simple* et le *non-transactionnel complexe* n'est pas aussi claire. Intuitivement, on sait cependant qu'un système de gestion manufacturière est plus complexe que la gestion d'inventaire d'une quincaillerie.

Les AGL (Ateliers de Génie Logiciel)

Le génie logiciel, c'est l'informatisation de l'informatique (certains disent même que c'est la revanche des utilisateurs !), c'est offrir un support matériel et logiciel aux informaticiens pour les aider à analyser, concevoir, réaliser, tester... Il est appelé aussi *l'ingénierie des systèmes d'information*.

DÉFINITION

L'«atelier de génie logiciel» peut être défini comme :

- un ensemble d'outils logiciels pour **l'informatisation des systèmes d'information** (en gestion, mais les domaines scientifique et industriel sont concernés aussi par la notion). **Chaque outil sert de support pour une ou plusieurs tâches** dans le cycle de vie. L'objectif final est bien de construire (génie) des systèmes opérationnels. Certains incluent aussi les matériels.
- un ensemble cohérent de composants : au sein d'une même structure technique, **les outils possèdent des interfaces de dialogue semblables et souvent une base de données communes**, et communiquent entre eux. L'A.G.L. s'appuie sur des préconisations de méthodes doit être distingué des outils de génie logiciel épars et isolés.

Selon les auteurs, praticiens et vendeurs, des expressions voisines sont employées :

- le terme anglo-saxon CASE (**C**omputer **A**ided **S**oftware **E**ngineering) peut être traduit par Ingénierie du logiciel assistée par ordinateur. Il est beaucoup employé dans les dénominations d'outils;
- l'abréviation O.G.L. pour **outil de génie logiciel**, de nombreux fournisseurs parlent maintenant «d'outils CASE»;
- A.I.G.L. pour **atelier intégré de génie logiciel** : cette expression souligne le fait que tous les composants travaillent autour d'un même référentiel;
- le terme anglo-saxon I-CASE pour *Integrated-CASE* : c'est l'idée d'intégration et de cohérence d'un ensemble d'outils qui est soulignée ici, c'est une bonne traduction d'A.I.G.L.;
- S.E.E. pour **Software Engineering Environment** : on parle aussi en français d'environnement d'ingénierie du logiciel et d'environnement de **génie logiciel** (E.G.L.);
- l'acronyme G.L.A.O. (**G**énie **L**ogiciel **A**ssisté par **O**rdinateur).

STANDARDISATION DES A.G.L.

Plusieurs initiatives de standardisation ont été prises ces dernières années, avec l'objectif de mettre au point une norme d'architecture des ateliers, pour harmoniser les façons de travailler et pour faciliter les communications entre outils (on parle aussi d'interopérabilité).

Quelques exemples peuvent être cités :

- S.F.G.L. (**S**ociété **F**rançaise de **G**énie **L**ogiciel) a été créée par Bull, Cerci, Euro-soft, Cisi, Steria, Sesa et Syseca, pour créer une plate-forme portable et multi-langage;
- P.C.T.E. (**P**ortable **C**ommon **T**ool **E**nvironment) provient du programme ESPRIT. Il a pour objet notamment la création d'une norme d'interfaçage pour assurer la portabilité des ateliers. Pour les données, il s'appuie sur la modélisation entité-relation. C'est un standard ISO depuis septembre 1993 (DIS ISO/IEC 13719). Le G.I.E. Emeraude en a fait la version commerciale.

Méthode et A.G.L.

Tout outil de génie logiciel repose sur une ou plusieurs méthodes, dont les préconisations sont « automatisées ». Cela peut aller d'un simple éditeur de textes ou de graphiques, au générateur de code ou de bases de données.

À l'inverse, certaines méthodes n'ont pas d'A.G.L. pour les mettre en oeuvre : on dit alors que la méthode n'est pas « outillée ».

Un service informatique peut d'ailleurs consacrer une partie de son budget à l'informatisation de ses normes et méthodes. Par exemple, la mise en place d'une norme de description des dossiers de fin d'étape peut donner lieu à la programmation d'un gestionnaire des plans de documentation, et de routines qui vérifient leur respect par une exploration des textes sauvegardés.

La mise en place d'un A.G.L. a pour objectif de favoriser l'obtention des avantages attendus déjà par l'emploi des méthodes, avec entre autres :

- la diminution des temps de conception et de réalisation,
- la coordination des équipes (partage de dictionnaires...),
- l'amélioration de la qualité des travaux (plus de rigueur, de formalismes, et surtout de dossiers bien présentés et propres !),
- le fait de rendre plus faciles les actions de maintenance.

Un plan de formation doit bien distinguer l'acquisition des méthodes de celle des outils : ce sont deux natures de formation totalement différentes et aussi indispensables. Une formation à un A.G.L. sans formation préalable à la méthode ne sert à rien. Une formation à la méthode sans formation à l'A.G.L. utilisé expose les personnes à bien des soucis de mise en oeuvre technique (problèmes de productivité à prévoir !).

Les types d'outils

En reprenant les définitions précédentes, il est possible de classer les outils qui participent à l'avancement des travaux des processus d'informatisation en trois niveaux.

LES MÉTA-OUTILS

Ils permettent de fabriquer soi-même son A.G.L. (un atelier sur mesure) définissant les spécifications de sa méthode. Le préfixe *méta* signifie qui est à base des outils : on parle aussi de «méta-ateliers». Le concepteur d'A.G.L. définit :

- le dictionnaire et sa structure (les objets, leurs rubriques et leurs relations),
- ses modèles de représentation des systèmes d'information, avec ses propres dénominations,
- ses formalismes (c'est-à-dire les dessins),
- des rapports de contrôle qualité (définition des règles de vérifications sur les modèles),
- les procédures et formats d'exportation des spécifications (déchargement du dictionnaire).

LES ATELIERS DE GÉNIE LOGICIEL

L'A.G.L. est une structure d'accueil commune, ou avec des interfaces de communication de base. C'est un système d'outils offrant un support plus ou moins complet du processus d'informatisation. Par exemple, certains éditeurs offrent des chaînes de production de logiciels (le terme anglo-saxon usité est *Software factories*) qui représentent un cas particulier d'A.G.L. consacré à la conception technique et à la réalisation des applications.

Le caractère de plus en plus complet des ateliers et les espoirs de voir un jour émerger une norme, font penser à certains auteurs qu'il sera possible de pouvoir interchanger des outils au sein des structures d'accueil de type A.G.L. (l'expression anglo-saxonne est «plug in - plug out»).

Cette évolution vers des environnements interchangeables se réalise déjà unitairement dans certains services informatiques, mais au prix d'efforts de développements d'interfaces et de ponts de toutes sortes, et pas toujours prévus dans les budgets (voir *Figure 1*). Cela est fait aussi avec l'arrière-pensée de se garantir contre le risque de dépendance

trop forte envers un fournisseur unique. Certaines normes d'échange (par exemple CDIF, **C**ase **D**ata **I**nterchange **F**ormat) se dessinent, mais aucune n'a encore été adoptée.

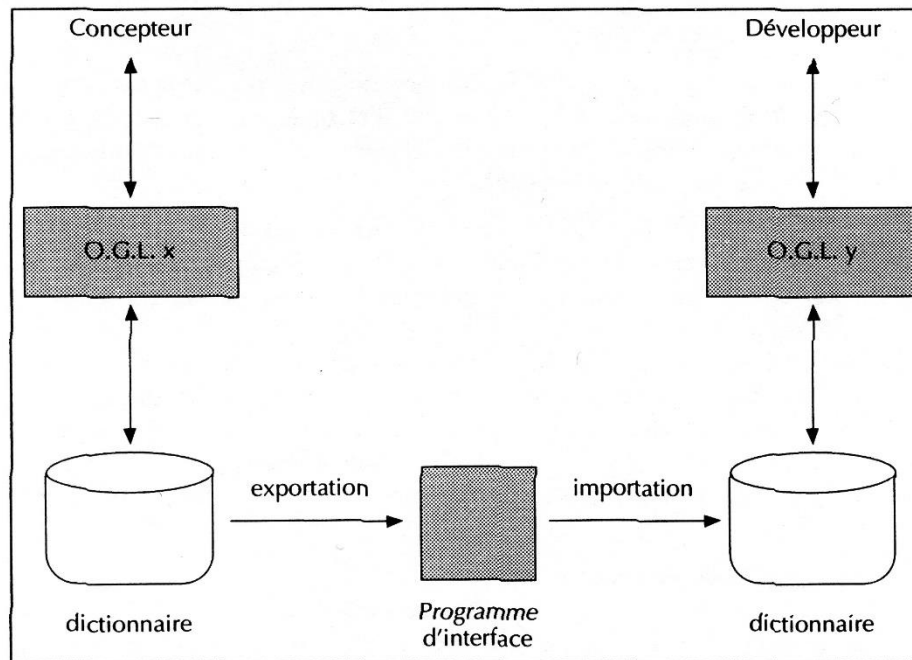


Figure 1 : interfaçage d'outils de génie logiciel.

LES OUTILS DE GÉNIE LOGICIEL

Pour montrer la différence entre un A.G.L. et un ensemble d'O.G.L., l'exemple des logiciels intégrés sur micro-ordinateur peut être pris : les intégrés proposent un traitement de texte, un tableur, un gestionnaire de fichiers, un outil graphique, un logiciel de communication... et tout ceci dans un même ensemble avec des menus communs. L'autre solution est d'acheter des outils séparés, en espérant que leurs normes de formats de fichiers soient communes pour bénéficier de partages de données. Il en est de même en génie logiciel. Mais cet exemple pourrait apporter une confusion : l'A.G.L. n'est pas qu'un seul logiciel (qu'un seul programme pour simplifier). Il est une structure qui résout surtout les problèmes d'interfaces.

Les O.G.L. ou les A.G.L. (en partie ou en totalité) doivent être accessibles sur les postes de travail des intervenants dans le processus d'informatisation, pour les travaux qui les concernent. Cela impose au Responsable des Méthodes de bien connaître les rôles de chacun et de proposer en conséquence les outils adéquats. Dans une Organisation, le génie logiciel aura en fait pour définition la portée que voudra bien lui donner ce responsable.

Comme le montre la *Figure 2*, les O.G.L. peuvent se séparer en deux familles par rapport à la conduite d'un processus d'informatisation :

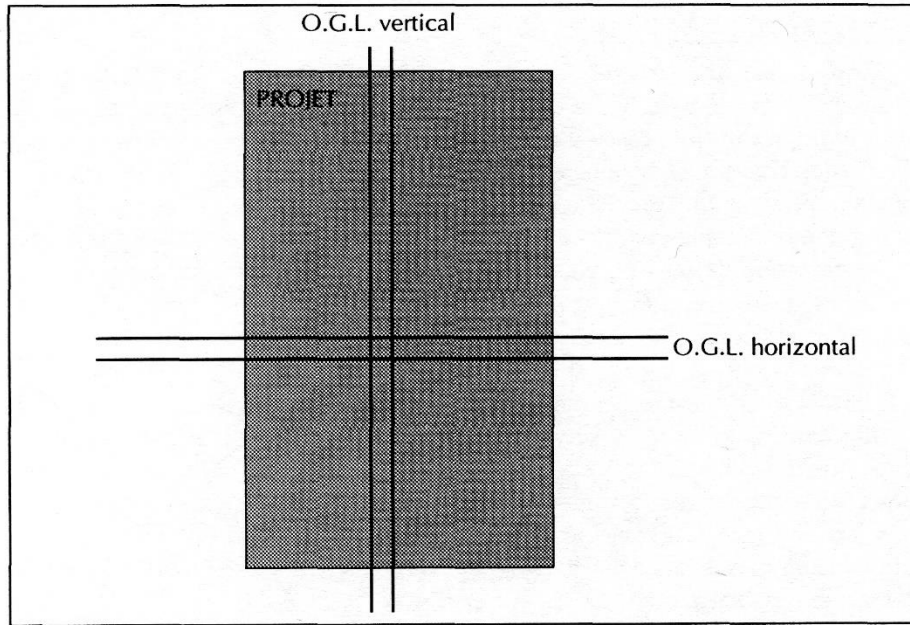


Figure 2 : les types d'outils de génie logiciel.

- les « verticaux » : ils sont utilisés tout au long des projets. Cela peut aller du simple outil de documentation des dossiers (du schéma directeur à la description-utilisateur d'un programme !), jusqu'à un outil d'ordonnancement qui servira à piloter chaque étape.
- les « horizontaux » : ils servent pour une tâche spécifique. Il s'agit par exemple, d'un compilateur, d'un analyseur de code, d'un outil de dessin d'un modèle de représentation du système d'information, ou d'un générateur de structures de données... On distingue ainsi les « upper CASE » qui supportent les phases en amont du cycle de développement, et les « lower CASE » pour les tâches de réalisation ou de maintenance technique. Mais la frontière entre ces deux familles est délicate à tracer : elle se situe autour de la conception technique du logiciel.

L'A.G.L. est l'association d'outils verticaux et horizontaux au sein d'une même structure constituée par un fournisseur ou par un service informatique d'une Organisation (on parle dans ce cas d'A.G.L. « maison »).

La *Figure 3* montre la différence entre une démarche de travail dans un environnement de génie logiciel composé d'outils non intégrés (cas 1), et celle autour d'un atelier (cas 2).

Dans le cas 1, les concepteurs produisent des résultats sur des supports spécifiques (papier ou magnétique). La démarche est linéaire, et se caractérise par des allers et retours éventuels. Le développeur traduit les spécifications en langages codés pour la machine. Il peut être assisté par un générateur, mais sa présence est indispensable pour mettre en oeuvre les outils techniques.

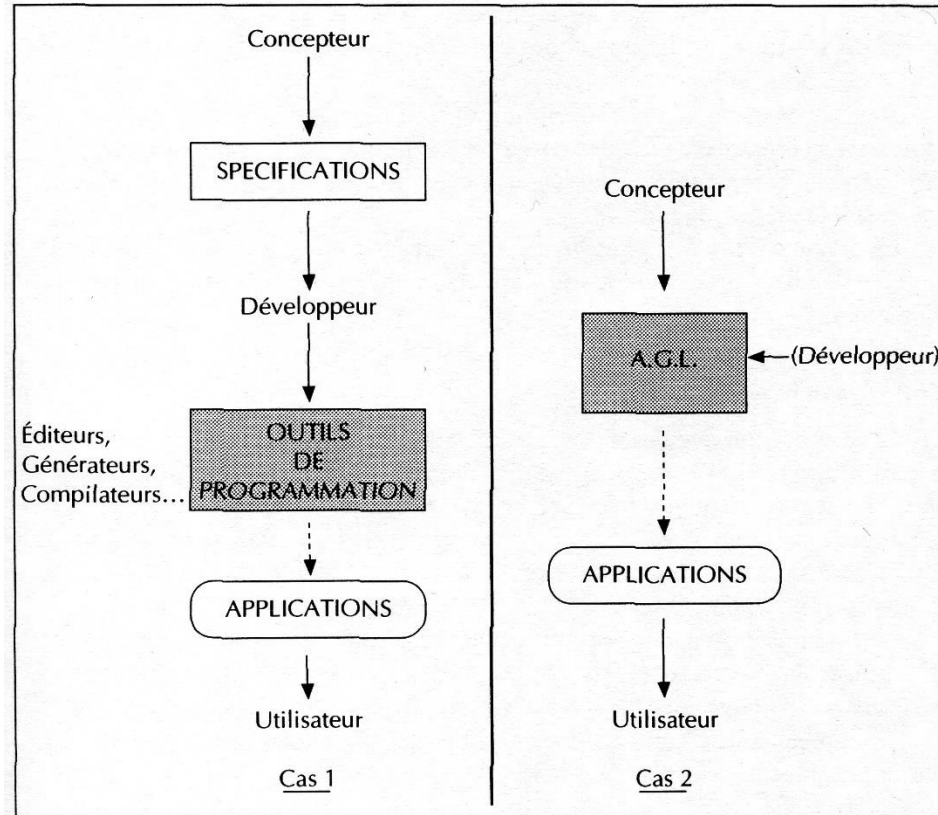


Figure 3 : démarches de travail sans et avec un A.G.L.

Avec le cas 2, l'A.G.L. centralise (par son dictionnaire) tous les travaux, et le rôle du développeur peut en être diminué grandement, car les outils de génération sont intégrés à ceux de conception. La distinction entre outils «upper CASE» et «lower CASE» est plus difficilement faisable ici. Des structures de bases de données et même des programmes (avec des squelettes pré-enregistrés) peuvent être automatiquement produits par l'interprétation de paramètres saisis par le concepteur.

Les fonctions

Les décideurs en informatique trouveront ici les fonctions utiles pour mettre en oeuvre les méthodes d'une façon assistée par ordinateur¹.

Cette présentation, sous forme de liste de contrôle (les anglo-saxons diraient *check-list*), est faite pour leur donner les préoccupations dans un processus d'informatisation. Elle les aidera dans la phase de choix d'un A.G.L. ou d'un O.G.L., pour vérifier que l'outil candidat possède bien la capacité à assurer les fonctions utiles.

¹. Dans le langage courant, les informaticiens emploient le terme «fonctionnalités» au lieu de parler plus simplement de fonctions d'un outil.

Dans cette optique, une description des A.G.L. n'aurait pas été aussi profitable car ils comportent tous des ensembles différents de fonctions, et sont donc difficilement comparables. De plus, ils évoluent très vite, et cette étude aurait été obsolète en peu de temps.

En revanche, pris sous l'angle des fonctions, en étant le plus exhaustif possible, le décideur pourra comparer des A.G.L. en vérifiant qu'ils offrent ou pas leur support.

RECUEIL DES BESOINS

La formulation des besoins d'un utilisateur demande une étude précise de la part d'un analyste non spécialiste du domaine à informatiser. Sa traduction sous formes de langages structurés et de graphiques doit s'accompagner de techniques d'entretien et de conduite de réunion. Une analyse syntaxique et sémantique des phrases peut être utile comme le recommande certaines méthodes, mais peut s'avérer très vite fastidieuse.

La constitution de dossiers de recueil, sous une forme formalisée, facilite l'analyse et la rédaction d'appels d'offres. À ce niveau, un bon traitement de texte, accompagné de tables des matières des divers documents peuvent faire l'affaire.

Quelques connaissances en psychologie sont aussi utiles, mais peu automatisables !

ESTIMATION DES CHARGES ET DÉLAIS

Le calcul des coûts d'un projet peut se faire selon plusieurs modèles. L'outil employé en dépend donc.

Il est utile de gérer des profils d'application pour obtenir facilement des prévisions en changeant des hypothèses de travail, par exemple des délais de réalisation de certaines étapes (simulations).

CONDUITE DE PROJET

Trois grandes fonctions sont proposées par ces outils : la gestion des tâches, des coûts et des ressources. Ils aident pour :

- la tenue de planning : création et réactualisations à l'aide de tableaux ou de graphiques, avec un ordonnancement sous forme PERT ou GANTT. Par exemple, la plupart des outils présentent des graphes avec en abscisse le temps et en ordonnée les intervenants. Des rectangles schématisent ainsi les tâches de chacun;
- la tenue des dossiers et documentations : ce sont les gestionnaires de documents associés à des traitements de textes, avec une prise en charge de la confidentialité (accès contrôlés en lecture, écriture, suppression);
- la communication de messages et de dossiers entre les intervenants : avec des messageries électroniques sur une machine ou entre plusieurs ordinateurs (les

dialogues ainsi différés font parler parfois de télé-réunions asynchrones !). Ce partage électronique d'informations tend à diminuer l'émission et la circulation de papiers;

- le suivi des événements : réunions, besoins de la maîtrise d'oeuvre, décisions de la maîtrise d'ouvrage sur des fonctions...
- la gestion des agendas : convocations aux réunions, demandes de rendez-vous...
- la prise en compte des aspects financiers, avec l'analyse de budgets, une comptabilité analytique par projet, la répartition des ressources...

CONCEPTION DES SYSTÈMES D'INFORMATION

Les outils proposent ici un support intimement lié aux méthodes employées.

Quelles que soient les préconisations suivies, il y aura un besoin de gérer des bases d'objets (textuels et graphiques) de description des systèmes. Une gestion fine des décalages entre la base de graphiques et du dictionnaire des objets est nécessaire.

Le dessin est un des éléments principaux de ces outils. Des palettes peuvent être proposées : l'utilisateur y sélectionne les formes (avec la souris) et les place dans les modèles, en changeant éventuellement leurs dimensions. Une fonction de couper/coller est généralement disponible pour faciliter les manipulations. Une fonction de WYSIWYG peut être proposée aussi : ce terme signifie ***What You See Is What You Get*** (en français *ce que vous voyez est ce que vous obtenez*). Elle visualise un graphique à l'écran exactement comme il sera imprimé. Elle est répandue dans les outils de traitements de textes, et offerte dans certains outils de dessin. La fonction de *zoom* sur les graphiques permet de l'obtenir par effet d'éloignement et de rapprochement.

Le concepteur commence son travail sur une page blanche, puis dessine son modèle, sauvegardant au fur et à mesure les objets dans le dictionnaire. Il peut ajouter des éléments propres au dessin, tels que des titres, des cadres, des couleurs... Cette façon de travailler est très souple et conviviale puisqu'il s'agit de procéder comme avec un crayon et une gomme.

Dans le cadre d'une étude comparative d'outils de conception, quelques critères sont regroupés ici :

- quels sont les niveaux de découpage des études (domaines, projets, applications...) et comment s'y définissent les modèles ? Peut-on faire plusieurs modèles de même type au niveau de découpage le plus fin ?
- quelles sont les consolidations proposées à chaque niveau de découpage ? Est-il possible de fusionner des modèles ?
- des contrôles automatiques et interactifs de cohérence des modèles sont-ils offerts pour signaler les anomalies dans la modélisation ?
- l'outil offre-t-il des langages structurés de spécification, en plus ou à la place de modèles graphiques ?

- est-il possible de créer ses propres graphismes, et ses propres types de modèles ?
- existe-t-il une fonction de génération de dessin ?
- peut-on faire plusieurs graphiques pour un même modèle ?
- les modifications des modèles peuvent-elles être faites indépendamment dans le dictionnaire et sur les graphiques (quelles répercussions) ?
- peut-on définir des vues sur les modèles ?
- peut-on relier des modèles entre eux par leurs objets (création d'hyperliens) ?
- quelles sont les performances de l'outil par rapport au nombre d'objets gérés dans un modèle ?
- une navigation sur le schéma de la méta-base (dans les modèles graphiques par exemple) est-elle possible ?
- l'outil est-il fidèle à la méthode d'origine ? Sinon, quels en sont les points de divergence ?
- l'outil gère-t-il la volumétrie, des commentaires et des descriptions sur les objets, des contrôles de valeurs... ?
- la génération de maquettes ou d'applications selon un standard de dialogue paramétré est-elle possible ?
- les dossiers de documentation peuvent-ils être composés spécifiquement (paramétrage) ?

Des critères plus spécifiques à la méthode implémentée pourront être ajoutés à cette liste. Par exemple avec la méthode MERISE, d'autres questions peuvent se poser :

- dans un MCD, les relations n-aires (avec $n > 2$) sont-elles gérées ? Les relations peuvent-elles porter des informations ?
- quels sont les liens possibles entre un MCD et un MCT ?
- quelle aide l'outil propose-t-il pour la normalisation du MCD ?
- le passage du MCD au MLD est-il automatique ou manuel ? Quelles représentations logiques sont proposées (CODASYL, relationnelle) ?

MAQUETTAGE ET PROTOTYPAGE

Ces deux termes peuvent prendre des significations légèrement différentes selon les Organisations.

Généralement, une maquette est considérée comme un défilement d'écrans qui donnent une idée de ce que sera la future application (sans accès à des fichiers ou à des bases de données), et un prototype comme une application en réduction qui fait des lectures et écritures de données.

Une maquette peut éventuellement autoriser des saisies de données (mais sans sauvegardes), des mouvements du curseur, l'emploi des touches de fonction, la création de variables de travail... Une fonction intéressante est de simuler une session soit pré-enregistrée, soit réellement interactive.

Maquettes et prototypes ont pour but :

- de valider des besoins d'utilisateurs,
- d'en provoquer une expression plus juste,
- et de lever d'éventuelles contraintes techniques.

Certains outils permettent de réutiliser ce qui a été développé pour les besoins d'une maquette ou d'un prototype, dans l'application réelle (gain de temps), en générant par exemple les structures de données associées.

GESTION D'UN DICTIONNAIRE

Les dictionnaires de données (ou d'objets) offrent un support pour la définition, la description, la recherche (avec de nombreuses sélections et des références croisées), l'impression (sous forme unitaire et de dossiers) et une génération éventuelle :

- d'objets prédéfinis : les entités, les données (et types de données), les structures de données (types de fichiers), les fonctions, les programmes, les écrans, les états, les applications, les utilisateurs, les mots-clés...
- d'objets spécifiques : quand une notion nouvelle doit être gérée, certains dictionnaires peuvent l'inclure dans leur méta-base. Dans ce cas, il est possible d'établir des relations entre ces nouveaux objets et ceux déjà existants : le méta-modèle peut être enrichi.

Les dictionnaires assurent la cohérence du système d'information et des systèmes informatiques. Ils sont d'ailleurs plus ou moins techniques : ils concernent les phases en amont et/ou en aval des cycles de vie comme le montre la *Figure 4*. Les parties conceptuelle et technique peuvent faire l'objet de bases et de sites distincts.

Pour les tâches de conception, les dictionnaires sont aussi le support d'objets graphiques qui représentent des modèles, en liaison avec un module de dessin. Dans ce cas, ils sont parfois intégrés à un atelier de conception.

Pour les phases de réalisation, une fonction de génération des structures de données est utile pour créer les types d'enregistrement selon différents formats de bases de données.

Les références croisées sont des utilitaires de recherches de références des objets dans d'autres objets. Elles permettent d'obtenir des rapports avec pour chaque objet tous les autres utilisés avec leur type. Elles peuvent s'appliquer aux modèles de conception comme aux objets techniques (programmes, sous-programmes...).

Pour les recherches, certains dictionnaires proposent des écrans de sélection d'ensembles d'objets, ou aussi une interface SQL (**S**tructured **Q**uery **L**anguage : langage structuré de requête).

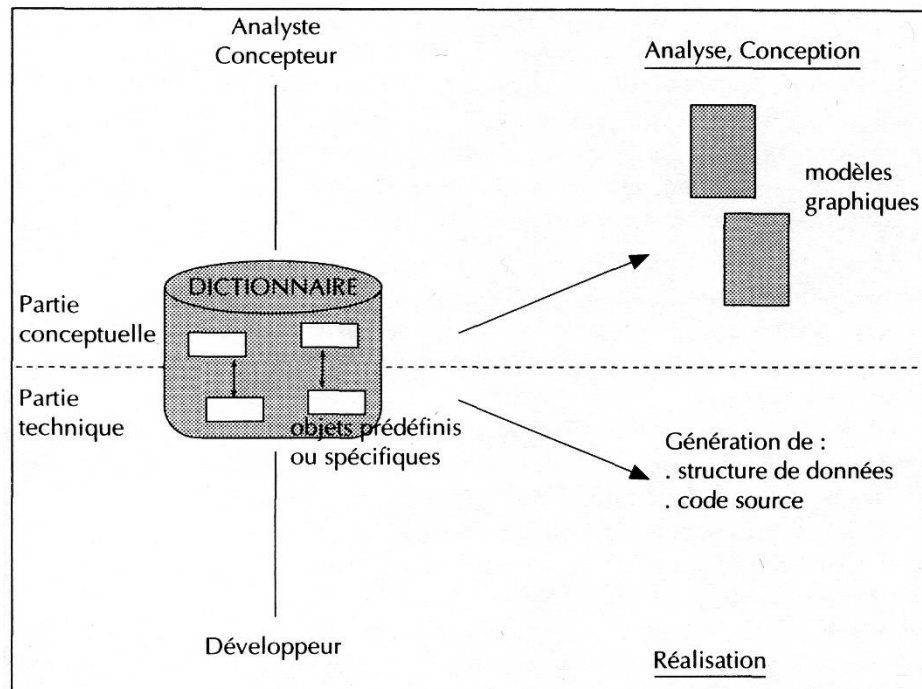


Figure 4 : les composants du dictionnaire.

Certains dictionnaires contiennent des éditeurs de modèles, de structures de fichier, de textes (documentations des objets), de graphiques (ils sont généralement couplé à un outil graphique), de code (ils sont intégrés alors à un environnement de développement), de règles de gestion, d'écrans et d'états... et tout autre éditeur dédié à un type d'objet spécifique. Ils peuvent être interfacés avec des générateurs. Certains proposent une fonction de génération de dossiers qui composent les descriptions des objets avec des tables des matières, des têtes de chapitre...

Dans les architectures atomisées de matériel, il est nécessaire de procéder à des consolidations des différentes bases du dictionnaire, pour en contrôler la cohérence et pour mettre à jour toutes ses parties.

Le terme *encyclopédie* est utilisé dans certains services informatiques pour désigner le dictionnaire.

MANIPULATION DE BASES DE DONNÉES

Ces outils permettent :

- d'effectuer des requêtes de création, de mise à jour et de suppression,
- à l'aide de tableaux, graphiquement, ou avec un langage non procédural et des mécanismes de génération,
- dans toute base de données constituée.

Cela est utile pour créer des jeux d'essai, ou pour vérifier la bonne exécution de composants logiciels.

Ils se présentent sous la forme d'outils de type infocentre, mais à l'usage des informaticiens.

PROGRAMMATION ET TESTS

Du simple éditeur au générateur le plus évolué... la diversité des outils doit être considérée ici surtout par rapport aux langages employés. De même pour les tests unitaires et d'intégration (techniques et fonctionnels) : les exécutions pas à pas, les points d'arrêts, l'observation des zones en mémoire, les tests sous conditions, et autres techniques dépendent des outils de programmation.

Les éléments suivants sont à considérer :

- l'aide à l'écriture, avec des fonctions d'édition plus ou moins élaborées : textes libres ou contraints, multi-fenêtrage, commandes paramétrables, polices de caractères, glossaire, «indentation automatique», vérification de syntaxe (navigation de faute en faute)...
- la génération de code source, avec des squelettes de programmes, et des paramètres de génération récupérés du dictionnaire ou saisis,
- l'assemblage (de composants logiciels),
- l'édition de lien (résolution des références),
- l'interprétation,
- la pré-compilation (traduction de certaines parties du code pour le rendre homogène),
- la pseudo-compilation (création de modules non compréhensibles par l'homme mais pas exécutables d'une façon autonome),
- la compilation : de plus en plus souvent les développeurs travaillent sur des stations autonomes pour obtenir des temps de compilation plus courts que sur les «mainframes»; l'outil peut offrir de contrôler l'ordre des compilations, d'interdire les recompilations, et d'assurer la cohérence avec les codes sources,
- l'optimisation d'instructions pour l'amélioration des performances,
- la qualimétrie (mesure de la complexité et des anomalies syntaxiques),
- la gestion des versions des modules sources et exécutables, avec des bibliothèques de sources et de fichiers binaires, sur les différents environnements (de développement, de tests, et d'exploitation); l'outil peut gérer l'historique des maintenances, avec des numéros de version et des indices d'évolution,
- la gestion des configurations de logiciels (ensemble de lots de composants applicatifs et «systèmes»),
- la mise au point des programmes et la conduite des phases de tests, avec éventuellement des mécanismes de localisation du point courant, de trace d'exécutions, de contrôle de la propagation des exceptions, de lecture des données internes ou échangées, de liste des points d'arrêt, de téléchargement et de télé-mise au point sans quitter la machine hôte, d'analyse de la mémoire

(accès à la pile des appels...), d'exécution de code spécial pour test, de restructuration de code...

- la création et la gestion (à différentes phases des tests) des jeux d'essai.

GESTION DE LA QUALITÉ

Selon le Plan d'Assurance Qualité défini au sein du service informatique ou même au niveau de l'Organisation, une gestion des critères de qualité et un suivi des actions de contrôle sont nécessaires.

La qualimétrie des logiciels est une activité de gestion de la qualité aux phases de conception technique et de réalisation. Un lien plus ou moins étroit et automatisé avec le dictionnaire peut être créé, tel que la génération des résultats d'un contrôle dans la documentation de l'objet concerné.

INTERFAÇAGE DES COMPOSANTS

L'emploi d'outils de génie logiciel d'origines diverses oblige à développer des programmes pour faire communiquer certains éléments d'un atelier composite.

Au plus simple, il s'agira de programmes de déchargement de données, dans certains formats puis de chargement. Certains «ponts» offrent ces fonctions avec la prise en compte d'un nombre plus ou moins élevé de formats. Les fournisseurs offrent généralement avec leurs produits des modules d'interfaçage (au moins pour les importations !).

MODULE DIRECTEUR

C'est un logiciel qui gère les enchaînements entre des modules, en assurant le passage de paramètres spécifiques à ces modules ou généraux à l'application. Il décharge les développeurs, en partie ou en totalité, de la gestion du dialogue homme/machine et fonction/fonction (au sens tâche programmée).

En outre, il peut prendre en charge :

- la gestion des menus, avec les hiérarchies d'options offertes par «profil» d'utilisateur (d'où une gestion d'autorisation). Cela rend possible la création de menus dynamiques où seules les options auxquelles a droit un utilisateur sont affichées;
- l'ouverture, l'interruption, la suspension, et la fermeture des sessions;
- la fonction d'assistance aux utilisateurs : active (recherche de valeurs de sélection) ou passive (affichage de documentations en provenance du dictionnaire de données par exemple);
- la gestion de messages de service.

Des moniteurs de transaction, des environnements de développement (surtout les langages et outils de quatrième génération) peuvent comporter à la base certaines de ces fonctions.

Le développeur est assisté par une application d'administration (un poste d'administrateur peut être confié à un intervenant) pour définir les applications, les fonctions, et autres données utiles en relation éventuelle avec un dictionnaire.

TRANSMISSION DE DONNÉES

Il est nécessaire de transférer des fichiers dans le cas de sites différents entre :

- deux environnements de développement,
- un environnement de développement et un autre d'homologation,
- l'environnement des tests finaux et celui d'exploitation.

La *Figure 5* montre un exemple de configuration.

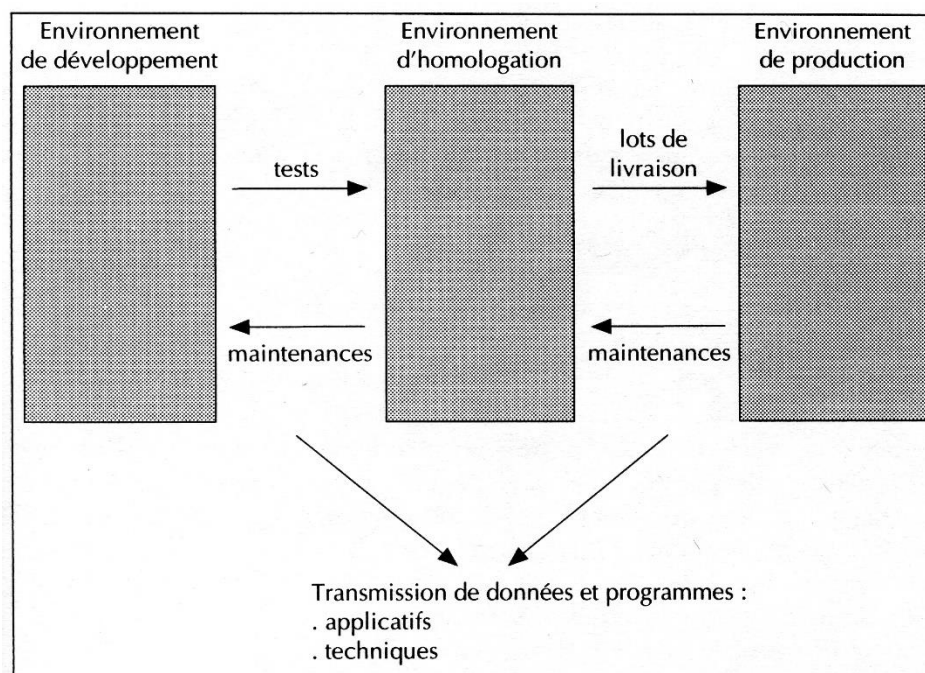


Figure 5 : les environnements de travail.

Les fichiers sont applicatifs ou techniques. Ils contiennent des données ou des traitements.

Cette communication doit être organisée précisément pour gagner un temps précieux pendant les phases de développement et de test, ou pour des mises en production efficaces (une notion de lot de livraison peut y être attachée). Des mécanismes d'émission (avec attente d'accusés de réception, et de retransmission automatique) et de réception (avec des archivages définis) sont généralement proposés.

MISE EN PRODUCTION

La rédaction de dossiers de production est considérée comme une corvée par certains développeurs, alors qu'elle est indispensable à une exploitation efficace des applications. Elle dépend du degré d'automatisation des tâches de production.

La création des procédures et autres paramètres pour la mise en production, les migrations, les enchaînements, et les exécutions peut être assistée (avec des mécanismes de génération). Il faut penser aussi à la gestion des reprises d'exécution en cas d'incidents.

Les services de production possèdent le plus souvent des normes précises pour tous ces éléments.

FORMATION DES INTERVENANTS

Les logiciels d'E.A.O. (**E**nseignement **A**ssisté par **O**rdinateur) se développent de plus en plus, et représentent aujourd'hui un marché important. On les trouve aussi bien sur micro-ordinateur que sur «mainframe».

La formation concerne :

- les informaticiens : ils doivent mettre à jour fréquemment leurs connaissances dans un domaine en constante évolution (cela est particulièrement vrai pour le génie logiciel). Une assistance plus ou moins élaborée et automatisée peut être nécessaire pour une compréhension rapide et complète de nouvelles techniques. La qualité des futures applications en dépend.
- les utilisateurs : la formation est ici un gage du succès de l'acceptation des systèmes développés. Certains logiciels d'E.A.O. permettent de reproduire à l'identique les écrans d'une application, en simulant des sessions de travail, en corrigeant les manipulations effectuées, et en gérant l'acquisition des connaissances.

ADMINISTRATION DES OUTILS DE GÉNIE LOGICIEL

Tous les outils et leurs adaptations éventuelles doivent être gérés en termes d'inventaire, de gestion des utilisateurs et des installations avec leurs versions sur les machines... et de partage des ressources.