La programmation complexe

par Christophe Garnier

Date de publication : 01/09/2005

Dernière mise à jour :

La complexité d'un programme apparaît à un développeur dès que sa perception lui pose des problèmes de compréhension. La taille du projet est certainement un facteur important mais la complexité perçue dépend aussi de l'expérience et du savoir acquis par le programmeur. La maîtrise de cette complexité a un caractère général et ne concerne pas uniquement la programmation. Le but de ce document est de s'intéresser à la manière dont on peut la gérer. Pour cela, on se propose de définir un modèle de comportement basé sur une Analyse Systémique que l'on définit à partir de quelques règles. Une section est consacrée au domaine de la programmation et une autre conclut ce document en présentant le concept d'informatique complexe. Téléchargement PDF de l'article (15 pages)

I - Introduction

La complexité d'un problème informatique n'est pas perçue de la même façon par les programmeurs. La plupart des projets possèdent un aspect complexe et plusieurs facteurs en sont responsables:

- la "taille" du projet,
- la "complexité" du système étudié (comportement, structure),
- l'hétérogénéité du projet (plusieurs systèmes d'exploitation, intégration de bases de données, systèmes en réseaux, ...).
- ...

Ces facteurs sont en quelque sorte intrinsèques aux projets et participent d'une "manière fixe" à la complexité perçue. Pourtant, cette dernière est aussi d'essence subjective dépendant directement du "savoir" et de la compétence du modélisateur Du fait de la densification des moyens et des outils mis à disposition, la maîtrise d'un projet informatique peut nécessiter un savoir de plus en plus vaste dans différents domaines (logiciels, langage(s), code, APIs, ...) . On peut considérer que la connaissance et la complexité perçues d'un projet s'opposent de manière concurentielle. Par exemple, un spécialiste possède a priori (?) la connaissance nécessaire pour résoudre un problème spécifique à son domaine de compétence. Son expérience lui permet d'estimer la complexité de manière plus favorable que ne le peut un modélisateur sans connaissance particulière.

Toute stratégie pour la maîtriser s'appuie nécessairement sur des phases d'analyse (recherche de la compréhension) et d'apprentissage, phases que l'on peut essayer d'"optimiser" en les organisant en fonction d'un ensemble de règles. On se propose de définir une méthodologie permettant de gérer la complexité d'un problème de la manière la plus efficace. Pour cela, on définit un **modèle de comportement général** possédant deux caractéristiques. Le modèle inclut:

- un système de gestion de la connaissance (le savoir);
- un système de raisonnement dont les principes sont définis par l'**Analyse Systémique**.

Le propos de ce document est de définir ce modèle de comportement en s'appuyant sur la définition d'une **Analyse Systémique**. On consacre par ailleurs une section à son exploitation dans le cadre de la programmation.

II - Un peu de théorie

A - L'Analyse Systémique

Il existe plusieurs façons de présenter l'Analyse Systémique.

L'une d'entre elle consiste à l'opposer aux quatre préceptes d'une analyse dite "classique" proposée par R. Descartes dans son "discours de la méthode". Au contraire de son homologue, la Systémique est particulièrement adaptée à l'analyse des systèmes complexes et de leurs cortèges d'interactions multiples et entrecroisées:

- au précepte d'évidence, elle oppose le principe de pertinence basant la modélisation en fonction de "finalités explicitables" de l'Analyseur;
- au précepte d'exhaustivité, elle oppose le principe d'agrégativité mettant en oeuvre le principe de pertinence pour la création d'une représentation simplificatrice du système étudié;
- au précepte de causalité, elle oppose le principe téléologique portant sur la recherche des finalités des caractéristiques du système étudié;
- au précepte de division des difficultés, elle oppose le principe de globalité portant sur la perception de systèmes en interaction dans leur environnement. Ce principe met en valeur le fait qu'un système n'est pas seulement réductible aux éléments qui le composent.

On peut faire les remarques suivantes.

1 - Le principe de pertinence

La particularité de ce principe est de donner une place privilégiée au modélisateur en lui "accordant le droit" de rédiger ses objectifs. Ces derniers permettent d'une part d' "amorcer" les processus de modélisation, et d'autre part, de le guider en fournissant un "cadrage" de ses actions de modélisation.

La Systémique considère les objectifs de modélisation comme des informations faisant partie intégrante des modèles construits. Par exemple, lors de l'analyse de programmes, il est important de garder à l'esprit que ceux-ci ont été développés en fonction d'objectifs (in)conscients. La recherche de ces derniers peut fournir l'équivalent d'un "fil rouge" pour la conduite de l'analyse des programmes mais aussi lors de leurs constructions. Dans ce dernier cas, le modélisateur "s'autoguide" pendant son raisonnement.

A la lumière de la description précédente, la complexité perçue peut correspondre à une estimation de la part du modélisateur de sa propre capacité à pouvoir réaliser des objectifs désignés. Sa maîtrise se conçoit dans un premier temps dans la **claire définition** de ces objectifs. Une mauvaise "vision" de ces derniers ne peut entrainer qu'une complexité accrue tandis que le contraire modifie positivement cette perception.

2 - Le principe d'agrégativité

Ce principe fait intervenir la notion de **visibilité restreinte** largement répandue dans les langages de programmation (notions de "classe", "objet", ...). La représentation simplificatrice qui en découle permet une meilleure compréhension des caractéristiques modélisées du fait du plus petit nombre d'informations présentées. Utilisée à bon escient, elle peut permettre au modélisateur d'améliorer sa "vision", donc sa maîtrise du modèle.

On peut remarquer que cette **visibilité restreinte** permet de mettre en valeur des caractéristiques spécifiques correspondant à un certain "axe d'analyse" du système étudié. Cet "axe" simule l'utilisation d'un **Langage** particulier. Celui-ci regroupe un ensemble de concepts **génériques** qui permettent au modélisateur de formuler un modèle descriptif.

Par exemple, le principe de pertinence correspond à un **Langage** regroupant un ensemble de concepts en rapport avec la notion de **projet**. Suivant le domaine dans lequel il est employé, ce **Langage** s'enrichit de notions particulières. Cependant, la plupart d'entre eux représente des concepts généraux (le projet, les objectifs, les phases de réalisation, durées estimées des différentes phases, date de réalisation, délai, ...).

L'idée de la **généricité** est alors intéressante car elle permet d'avoir accès à l'expérience (le savoir) du modélisateur. En effet, on peut considérer le fait que la connaissance d'un système peut se définir comme une combinaison d'un **minimum** de descriptions, celles-ci ayant été formées à partir de concepts de quelques **Langages** particuliers. Cette définition soulève cependant deux questions de taille qui sont traitées dans la suite du document:

- comment peut-on modéliser une expérience?
- comment peut-on rendre compte de la complexité d'un savoir?

3 - Le principe téléologique

A l'instar du principe de pertinence, le principe téléologique prend en compte les objectifs du modélisateur. Mais cette fois ci, son analyse correspond à une recherche des finalités des éléments du système étudié. Pour cela, le modélisateur peut se baser sur sa propre expérience, c'est-à-dire ses "axes d'analyse" (ses **Langages**) pour lesquels il possède une connaissance approfondie.

Par exemple, un programmeur s'attache à suivre une certaine démarche l'amenant à inclure dans son code des informations de type "structures de contrôle" ("for", "if", ...) possédant des finalités propre à leur fonction. Parallèlement, le programmeur agit à d'autres niveaux en leur **associant** d'autres finalités (principe de globalité).

4 - Le principe de globalité

Le modélisateur analyse suivant un nombre **limité** d' "axes" en percevant des organisations de systèmes agissant dans leurs **environnements**. Il possède alors une compréhension des caractéristiques du système étudié uniquement suivant ces "axes". Cependant, le système étudié étant supposé complexe, on peut aisément admettre que l'analyse peut se faire suivant d'autres "axes pertinents" dont il n'a peut-être pas la connaissance. Ceci implique que le modélisateur peut posséder une "vision" incomplète introduisant une certaine forme d'incertitude dans la compréhension des caractéristiques du système étudié. Cette incertitude permet d'expliquer en partie la perception complexe d'un système.

Ainsi, l'objectif de diminution de la complexité perçue correspond en partie à des actes de recherche et d'apprentissage de nouveaux "axes d'analyse" (des **Langages**) pertinents dans lesquels il effectue sa modélisation. Une analyse systémique se base sur l'application de deux principes:

- le **principe de décomposition** fournit le moyen de définir les éléments composant le modèle;
- le **principe de recomposition** permet de créer le modèle global.

Le principe de **décomposition** équivaut à une analyse selon des "axes d'analyse" (LS) fournissant au modélisateur une "vision orientée" des caractéristiques du système étudié. Avec l'aide des concepts modélisés de ces LS, il construit les modèles décrivant ces caractéristiques. La modélisation inclut par la suite des processus de **recomposition** menant à la réalisation du modèle global qu'il est nécessaire de **valider** suivant certains critères.

B - L'Information Systémique

La démarche adoptée est ici de définir un **Analyseur Systémique** lui permettant de maîtriser la complexité de ses projets. Comme il a été rappelé, ses qualités doivent s'exprimer à la fois dans le domaine de l'analyse et de l'apprentissage. Il se base par rapport à son savoir, sa "base de connaissance", pour analyser de la manière la plus pertinente possible.

En résumé, un **Analyseur Systémique** possède des mécanismes de raisonnement globalement fixés par l'Analyse Systémique et des capacités de mémorisation d'un "savoir". Il gère des informations d'un certain type que l'on retrouve à la fois dans son "savoir" et dans ses analyses.

On recherche dans un premier temps une définition de ces informations spécifiques.

Paul Valéry définit les mécanismes de raisonnement de la façon suivante:

"Nous ne raisonnons que sur des modèles".

En partant de cette affirmation, on peut supposer que le "savoir" et l'expérience sont en quelque sorte des "modèles" d'une certaine forme. Dans ce cas, un modélisateur expérimenté, possédant une connaissance suffisamment complète d'un domaine, "maîtrise" un ensemble de modèles associés. Au contraire, les problèmes perçus comme complexes correspondent à une connaissance limitée, c'est-à-dire, à un nombre **limité** de modèles pertinents. Un problème complexe ne sera alors maîtrisé que si l'on connaît un nombre **minimum** de modèles pertinents du système étudié.

On se base sur la théorie des Systèmes Complexes [1] pour compléter la définition. Cette théorie définit un modèle comme une combinaison d'informations particulières appelées **Informations Systémique** (IS) possédant trois composantes, le **Symbole** (1), la **Description** ou la **Désignation** (2), le **Processus** (3).

Le **Symbole** est la partie "visuelle" de l'**Information** (IS) et elle permet au modélisateur de manipuler cette dernière. En prenant l'exemple d'un langage de programmation , celui-ci met à la disposition du programmeur un ensemble de mots réservés représentant tous des concepts particuliers ("if", "for", "while", "int", "return", "class", "private", "public", ...). En les ordonnant, le programmeur modélise et crée une certaine **Description**. Tandis que le compilateur comprend le code selon son "langage"Les **Symboles** sont complètement spécifiés dans son cas., le programmeur travaille à différents niveaux d'abstraction. Ainsi, à chaque élément d'un programme, il peut associer une information d'un tout autre type. Par exemple, en langage objet, le programmeur peut implémenter une classe "User" pouvant représenter la définition **descriptive** d'un hypothétique utilisateur.

Enfin, le **Processsus** est la composante active de l'**Information** dont le but est la modification de l'état **descriptif**

de ces **Informations**. Dans le cas de la programmation, un programme compilé, c'est-à-dire traduit en langage machine, a son comportement complètement modélisé par un ensemble de **Processus** organisés hiérarchiquement en blocs de commandes (flux algorithmique). Ils ont pour tâche la modification de l'état de variables que le système informatique mémorise dans des structures de données particulières.

L'emploi de cette **Information** (IS) particulière revient à un **changement de paradigme** et une analyse va correspondre à définir un "univers de travail" dans lequel on postule :

- il n'existe que des **Informations Systémique** (IS) dans cet "univers";
- l'**Analyseur Systémique** raisonne et mémorise (**Savoir**) dans cet "univers".

Une autre particularité de l'**Information Systémique** (IS) est le fait qu'elle n'est pas directement accessible au modélisateur sauf au travers des **Symboles** la représentant. La compréhension que l'on peut avoir d'une **Information** est alors fonction du nombre et de la " qualité " (la pertinence) des composantes systémiques (**Description**, **Processus**) que l'on peut lui **associer**. Pour le modélisateur, des IS possédant de nombreuses associations de "bonne qualité" sont alors synonyme d'un niveau élevé d'expertise.

L'accessibilité indirecte de l'**Information** nous fait supposer que l'on ne peut pas la comprendre ni la décrire complètement. Une connaisance exhaustive ne peut représenter qu'une certaine limite théorique et la complexité perçue qui en découle ne peut être complètement abolie mais simplement réduite. Par conséquent, un modélisateur aura toujours la "liberté" de compléter la définition d'une **Information Systémique** (IS) en lui **associant** des descriptionsPeut-être contradictoire ou fausse! "passives" (Description) ou "actives" (Processus) selon un axe d'analyse particulier, un **Langage Systémique**.

Cet ajout modifie l'état du Savoir du modélisateur. Celui-ci correspond à:

- un ensemble d'**Informations** se "projetant" selon les **Langages Systémique** (LS) de l'**Analyseur Systémique**;
- une interconnexion des **Informations** suivant deux directions:
  - la première correspond aux modèles représentant des interactions entre les **Descriptions** et les **Processsus** d'un **Langage Systémique** particulier;
  - la seconde correspond aux composantes des **Informations** (IS) selon les L**angages**.

Ce document n'a pas pour objectif de caractériser de manière plus concrète (informatiquement) ce **Savoir**, le but étant ici de définir le comportement général d'un **Analyseur Systémique** supposé posséder un système opérationnel de gestion de sa connaissance.

C - Le modèle d'Analyseur Systémique

On se propose de baser le comportement de l'AnalyseurOu d'un groupe d'Analyseurs. sur une série d'opérations suivant, dans les grandes lignes, les principes énoncés précédemment. Pour cela, on construit un modèle d'Analyseur Systémique travaillant dans un certain "univers systémique":

- dans cet "univers", l'Analyseur manipule uniquement un seul type d'éléments, des **Informations Systémique** (IS);
- cet "univers" combine les propriétés d'un système de gestion de la connaissance (**Savoir**) et d'un système d'Analyse (raisonnement).

L'essentiel de son travail d'analyse se situe dans un "univers systémique". Il combine un ensemble d'opérations de recherche, de création, de modification et de destruction d'**Informations Systémique** (IS) mettant en oeuvre les principes énoncés:

- définition de ses objectifs (principe de pertinence);
- apprentissage et reconnaissance de son **Savoir** "utile" (évaluation de la complexité);
- analyse/modélisation des caractéristiques du système étudié (agrégativité et globalité).

On définit les règles suivantes.

1 - L'Univers Systémique

*Le modélisateur (ou l'équipe de modélisateurs) est un **Analyseur Systémique** manipulant **uniquement** des Informations Systémique (IS).*

L'analyseur accepte le **paradigme** de l' "Univers Systémique" en faisant l'hypothèse du caractère systémique de son savoir et de ses projets d'analyse. Dans les deux cas, les concepts manipulés se traduisent sous la forme d'Informations Systémique (IS) possédant les caractéristiques suivantes:

- un concept modélisé (IS) possède plusieurs définitions suivant certains **Langages Systémique** (LS) assimilés du modélisateur;
- la définition d'un concept modélisé (IS) est une combinaison de composantes passives (**Description**) et actives (**Processus**).

Le **Savoir** et le **système de raisonnement** d'un modélisateur sont deux aspects d'une même organisation, le second ayant besoin d'une partie du premier pour étudier les caractéristiques du système étudié suivant différents "angles" (sélection de LS pertinents). L'analyse mène par la suite à la création de nouveaux LS et de modèles s'intégrant dans le Savoir par des Processus d'apprentissage (association des IS du modèle avec celles de sa base de connaissance).

Dans cet "univers systémique", les LS sont un élément important du modèle de comportement et l'un d'entre eux, le **Langage Systémique de base**, est intéressant à mettre en valeur. Possédant le caractère le plus général, il représente un "axe d'analyse" des modèles selon les composantes de l'IS (**Processus** et **Description**). Dès lors, il peut servir de référence pour définir un certain "degré d'abstraction" des modèles (allant du plus général au plus spécifique).

2 - Le Savoir Systémique

*Le modélisateur organise son **Savoir** sous la forme d'un ensemble de Langages Systémique (LS) et de modèles constitués d'IS liées entre elles.*

Il est nécessaire que le modélisateur considère son **Savoir** comme une combinaison de **Langages Systémique** (LS) et de modèles assimilés. Ils représentent une connaissance **utile** à l'analyse de systèmes complexes pour la détermination des caractéristiques et des "axes d'étude".

Par la suite, l'analyse enrichit le **Savoir** par des Processus internes d'apprentissage des modèles et des LS créés

(intégration des IS par **association**).

3 - L'évaluation de la complexité

L'analyseur évalue la complexité du projet en étudiant de manière systémique ses objectifs de modélisation et les finalités des modèles. Il maîtrise la complexité en ayant pour objectif une "compréhension optimale".

L'analyseur conduit la détermination de ses objectifs en cherchant à construire un **modèle systémique** de son projet (confer section 3.4). Par ailleurs, il effectue sa modélisation en se basant sur une variante adaptée du "**Langage** (LS) de gestion de projet". Celui-ci lui permet de percevoir ses objectifs sous la forme d'une combinaison hiérarchisée de Processus "initiateurs" ("Processus-objectifs") de ses actions de modélisation.

Cependant, cette gestion ne mène pas nécessairement à une maîtrise d'un projet complexe, ces objectifs étant liés de manière intrinsèque à la modélisation du système étudié. Pour maîtriser cette complexité et posséder ainsi une connaissance optimale de son projet, l'analyseur a l'obligation (confer section suivante):

- de rechercher une bonne sélection des "Processus-objectifs",
- de décider du degré de finesseSoit il décide de ne pas poursuivre la décomposition d'un "Processus-objectif" (compréhension suffisante ou insuffisante nécessitant une phase d'apprentissage, ...), soit il poursuit la décomposition en définissant le "Processus-objectif" père (combinaisons de "sous Processus-objectifs").à accorder au **modèle systémique** de son projet,
- de documenter **clairement** ses objectifs,
- de valider leurs réalisations,
- de garder une trace des tâches effectuées (documentation de l'historique).

Ces deux types d'objectifs (objectif de modélisation et critère de "compréhension optimale") sont à prendre en compte dans les phases de validation des LS utilisés et des modèles créés.

4 - L'Analyse Systémique

*Le modélisateur analyse les caractéristiques du système étudié en recherchant par **Analogie** des LS, des modèles et des IS assimilés de son Savoir. Dans un second temps, il élabore les LS et les modèles des caractéristiques tout en les validant suivant des critères de pertinence et de compréhension optimale (assimilation optimale).*

L'objectif de maîtrise de la complexité d'un projet équivaut à une recherche de l'amélioration de la connaissance. La compréhension des caractéristiques d'un système est optimale si celles-ci s'assimilent à une combinaison de modèles se décrivant selon les **Langages** du modélisateur, et possédant des définitions simples et de faible complexité. La réalité est souvent différente et le modélisateur doit choisir ses **Langages** et construire ses modèles de telle sorte qu'ils puissent lui permettre une bonne intégration des IS dans son Savoir (critère de compréhension optimale des caractéristiques modélisées).

De manière générale, une analyse se définit par des phases imbriquées d'étude des caractéristiques, de construction de modèles et de validation des décompositions/recompositions, c'est-à-dire:

- des processus de définition des caractéristiques du système étudié devant prendre en compte leurs finalités dans le système étudié;

- des processus de construction de modèles décrivant les caractéristiques reconnues du système étudié selon des "axes d'analyse" (des LS assimilés) pertinents du modélisateur;
- des processus de validation des modèles en fonction des objectifs de modélisation et d'un critère de compréhension optimale.

a - Les caractéristiques du système étudié

L'analyse s'assimile dans un premier temps à une recherche par analogie des IS et des LS nécessaires à la définition des caractéristiques du système étudié. Un système se comprend d'autant mieux que celui-ci est "proche" c'est-à-dire **analogue** aux modèles et aux LS assimilés. A ce titre, l'expérience du modélisateur expérimenté joue ici un rôle primordial ayant acquis suffisamment de modèles de bonne "qualité" auxquels il peut se référer.

Une caractéristique se définit en étudiant sa finalité (passive ou active) dans son "environnement systémique". L'idéal serait pour le modélisateur d'avoir en sa possession un modèle prédéfini de cette caractéristique. Ce n'est généralement pas le cas et il se base sur son Savoir pour déterminer les éléments constituant ces "axes d'analyse" pertinents (LS). Pour cela, il peut travailler par **Analogie** pour rechercher les IS et les LS de son Savoir possédant une certaine affinité avec les caractéristiques du système étudié.

En considérant la définition de l'IS, trois types d'**Analogie** sont possibles.

Le premier type correspond à une recherche de concepts au niveau **symbolique** sans considérer de manière ciblée les composantes **Description** et **Processus** des IS. Au demeurant, cette recherche s'assimile grossièrement à une technique de type "brain storming".

Le second type d'**Analogie** correspond à une analyse selon la composante **Description**. Cette forme d'étude peut mener à un certain découpage dont le risque évident est de définir des décompositions de modèles inadaptées. D'un autre côté, ces décompositions peuvent correspondre à une représentation efficace du système, plus particulièrement si le système étudié est défini comme peu complexe.

Enfin, le troisième type **d'Analogie**, que l'on doit privilégier, se base sur l'analyse des composantes **Processus** de l'IS. L'idée consiste à rechercher les Processus analogue du système étudié. Ce type de recherche présente les meilleures propriétés pour pouvoir maîtriser la complexité et définir une modélisation de qualité.

La connaissance de LS pertinents est une nécessité pour la phase suivante de construction des modèles.

b - La construction et la validation des modèles

L'ensemble des LS et des IS sélectionnés permettent d'aborder de manière pertinente la construction du modèle des caractéristiques. Elle se traduit sous la forme d'un ensemble d'actions:

- l'analyseur identifie les LS pertinents et un ensemble de concepts correspondant aux caractéristiques du système étudié;
- l'analyseur adapte les LS sélectionnés au projet en ajoutant, modifiant ou détruisant des IS;

- l'analyseur compose son modèle en combinant les concepts modélisés des LS et en privilégiant l'étude des Processus. Le modélisateur peut d'autre part posséder des modèles analogues pouvant servir de base à la modélisation;
- l'analyseur recompose les IS du modèle en les associant selon les différents LS;
- l'analyseur valide les compositions des modèles par rapport aux objectifs fixés et en tenant compte d'un critère de compréhension optimale;
- l'analyseur assimile les IS et les LS des modèles créés en les associant avec ceux de son Savoir (apprentissage).

L'ensemble des IS retenues, organisées et reliées entre elles par l'intermédiaire de LS assimilés, représente le modèle du système étudié.

La maîtrise de la complexité propose de le valider suivant un critère d' "assimilation optimale", ce qui suppose une bonne compréhension de trois types d'information :

- les LS utilisés (bonne **association/intégration** des IS avec celles de son **Savoir**);
- les compositions d'IS décrivant les caractéristiques du système étudié selon un LS;
- les composantes des IS du modèle selon les LS utilisés.

Cette "compréhension optimale" d'une information est tout d'abord fonction de la **quantité** de concepts la décrivant. En effet, un analyseur est toujours confronté au problème du nombre limité d'informations appréhendable simultanément. La conséquence est qu'il est recommandé d'adapter les **Descriptions** d'un modèle en les construisant à partir d'un minimum d'informations.

L'autre aspect de la "compréhension optimale" d'une IS est la **clarté** des descriptions des trois types d'information (LS, composition d'IS, composantes d'IS). Une "connaissance optimale" ne demande pas à connaître de manière exhaustive les différentes parties d'un modèle mais de savoir déterminer une suite minimale d'actions (des Processus) permettant d'atteindre leurs descriptions.

La **lisibilité du symbolisme** utilisé (choix pertinents des noms des variables, symbolisme graphique adéquat, ...) est naturellement importante mais d'autres facteurs le sont aussi:

- choix des **LS utilisés**;
- qualité simplificatrice des **descriptions**Une **description externe** d'une IS représente un **environnement** dans lequel elle évolue. Simultanément, elle représente une **description** interne d'une autre IS selon un LS spécifique. Une **description interne** d'une IS représente la "projection" d'un ensemble d'IS selon un LS spécifique. externe et interne de l'IS étudiée.

Les deux derniers facteurs sont importants parce qu'ils doivent permettre de définir ce "chemin" optimal menant à la compréhension d'une caractéristique. Dans le cas où c'est possible, le modélisateur doit porter ses choix:

- sur les LS les plus connus ou acceptés (standardisation);
- sur les LS possédant un caractère simplificateur. Par exemple, il ne doit pas chercher à définir des LS regroupant un grand nombre de concepts disparates;
- sur un ensemble de modèles possédant une finalité claire et non complexe (description simplificatrice).

Le respect de ce critère de "compréhension optimale" peut mener à une complexification du modèle, c'est-à-dire, la construction de nouvelles **Descriptions**, voire la création de LS spécifique.

D - Application: la programmation complexe

On applique le modèle de Systémique développé dans le domaine de la programmation. Pour cela, on adapte les règles qui ont été précédemment définies pour la maîtrise de projets informatique complexes.

1 - L'Univers du programmeur

Selon le niveau d'expérience, le programmeur possède une vision "interne" plus ou moins élaborée des programmes. On suppose qu'il accepte le paradigme de l'univers systémique (et de la complexité des programmes) où toute les données gérées se modélisent ("se visualisent") sous la forme d'IS.

Quelles sont ces informations?

Les langages de programmation fournissent une grande partie d'entre elles où l'on retrouve certaines constantes:

- les informations sont contenues dans des fichiers organisés en répertoires;
- certaines d'entre elles modélisent des "flux algorithmique" ("if", "for", "while", ...);
- certaines intègrent des notions d'objets ("classes", "instances");
- ...

Ces types d'informations représentent des IS qui s'associent à un savoir spécifique à l'utilisation des langages de programmation. Mais en tant que "passerelles" de communication entre l'ordinateur et l'utilisateur, la plupart d'entre eux permettent aussi de gérer des concepts issus de "niveaux d'abstraction" plus proches de nos mécanismes de pensée, et donc a priori, plus facilement manipulables et assimilables.

Le **Savoir** et l'**Analyse Systémique** se nourrissent de ces deux types d'informations. Finalement, on peut distinguer trois catégories d'IS (et de LS):

- la première correspond aux informations liées à un langage de programmation.
- La programmation s'effectue à l'aide d'informations se situant dans une couche intermédiaire (seconde catégorie). D'un point de vue systémique, ces informations correspondent à un ensemble de Langages Systémique (LS) aux objectifs télélogique aisément discernables (gestion des fichiers, syntaxe interne des fichiers, gestion des classes, compilation et exécution, ...).
- Enfin, à l'autre extrémité du spectre des informations, la troisième catégorie correspond aux "domaines de modélisation" des caractéristiques du système étudié.

Les informations des "niveaux d'abstraction supérieure" (troisième catégorie) sont difficiles à cerner. Elles sont à la fois plus spécifiques et plus générales :

- elles sont spécifiques au domaine de modélisation du projet du fait des concepts manipulés;
- elles sont générales parce que la plupart des programmes implémentent une série de modèles non spécifiques à un langage et un domaine particuliers (analogie). Depuis quelques années, on assiste d'ailleurs à une standardisation de ces modèles ("templates") dont le but est l'amélioration de la qualité générale des programmes. Représentant un LS particulier, ils proposent des modèles d'organisations réputées efficaces pour la modélisation d'une certaine finalité.

2 - Le Savoir du programmeur

L'expérience d'un programmeur s'appuie sur une certaine connaissance des trois catégories d'IS, et sur l'utilisation d'une version du "Langage de projet" adaptée à l'informatique.

Une bonne connaissance lui permet de réagir efficacement en fonction des objectifs, l'effort nécessaire pour la réalisation d'un programme évoluant en fonction de l'expérience acquise. Il est alors importantPlus particulièrement dans le cas de projets complexes. d'investir le temps nécessaire pour l'assimilation de LS et de modèles d'organisation pertinents. Les sources d'informations sont généralement importantes et on peut privilégier l'étude des "tutorials" ou la lecture de livres spécialisés, livrant une connaissance organiséeSelon plusieurs LS. du domaine de modélisation (meilleure assimilation).

3 - Les objectifs du programmeur

D'un point de vue systémique, que ce soit lors d'une programmation (modélisation informatique) ou lors de l'analyse de codes, l'objectif du programmeur est la construction d'un modèle représentant **sa connaissance du système complexe étudié**. Le programmeur a l'obligation de définir un modèle de bonne facture devant permettre une compréhension optimale de ses caractéristiques.

Les **Informations** caractérisant directement le programme et celles s'associant aux objectifs sont des IS où les premières sont les résultats des opérations initiées par les secondes (LS de projet). Dans la pratique, un programmeur ou un groupe de programmeurs suit un cycle de développement dont les différentes phases sont plus ou moins standardisées. On retrouvera comme informations essentielles à la maîtrise de projets complexes :

- des informations fournissant l'historique des objectifs réalisés;
- des informations dédiées aux objectifs en cours.

Dans le dernier cas, la connaissance du projet complexe sert de référence à la planification des futurs objectifs. En particulier, l'estimation de la durée nécessaire à une implémentation est intéressante à effectuer quitte à affiner son jugement ultérieurement. Elle peut servir de test pour définir un "degré" de connaissance du projet en obligeant le programmeur à déterminer les différentes étapes de réalisation. Si celui-ci maîtrise son sujet, l'estimation sera d'autant plus précise, et d'autant plus incertaine dans le cas contraire.

4 - La programmation

Un projet complexe de programmation se décompose suivant un ensemble de Langages Systémique (LS), chacun d'entre eux s'intéressant à un aspect particulier du système étudié. Comme elle a été défini auparavant, l'analyse systémique propose trois types de processus:

- définition des caractéristiques du système étudié,
- implémentation en fonction de la connaissance du programmeur (LS et modèles),
- validation du modèle développé suivant les objectifs fixés et un critère général de compréhension optimale.

a - Les caractéristiques du système étudié

Le programmeur recherche dans un permier temps les IS et les LS lui permettant d'implémenter les caractéristiques du système étudié. Pour cela, il recherche des modèles et des LS assimilés ayant un rapport avec le problème posé (analogie).

La découverte des LS et des concepts utilisés mène naturellement à des actions d'apprentissage (plus ou moins longue...) et des actions de modélisation des caractéristiques reconnues dans le système étudié.

b - La construction des programmes et la validation des modèles implémentés

Le choix des LS est important parce qu'ils servent de support à l'implémentation des caractéristiques reconnues du système étudié. Ils définissent un « environnement de développement » dans lequel le programmeur élabore ses modèles. Les concepts reconnus et les LS sélectionnés lui permettent de modéliser selon des LS (on préviligie la modélisation selon la composante Processus des IS), la création du modèle global se faisant ensuite par association des IS entre elles.

Comme on peut s'en douter, la manière de programmer dépend grandement du langage utilisé du fait de l'éventail des concepts manipulés et de la structure imposéeIl est par exemple déconseillé de faire de la programmation "objet" avec le langage C (inversement, il est aussi déconseillé de programmer de manière procédurale avec C++).. Mais, du point de vue de la systémique, ces langages se composent de LS dont les concepts se modélisent:

- selon la composante Processus de l'IS (méthodes de classe, fonctions, procédures, blocs et lignes de commande, structures de contrôle, ...),
- selon la composante Description de l'IS (classe, attribut de classe, structure, int, char, pointeur, String, ...).

Au-delà du fait que les programmes doivent être validés selon des objectifsIl existe par ailleurs des solutions permettant la validation de ces objectifs (Extreme Programming, Junit, ...). de modélisation, le programmeur doit aussi évaluer leurs qualités en fonction du critère de "compréhension optimale" exprimant un certain degré de connaissance. Du fait que l'on se place sur le plan de la connaissance, il est alors justifié de prêter une attention particulière à certains détails de la programmation:

- il est évident qu'il est nécessaire de décrire le programme (et les concepts manipulés) de manière plus ou moins détaillée, que ce soit sous une forme graphique (ex: le langage UML) ou textuelle;
- en tant que "partie visuelle" des concepts manipulés (IS), le Symbolisme utilisé doit faciliter la reconnaissance des concepts modélisés. Ceci concerne tous les noms utilisés dans le programme (variables, paramètres, méthodes, fonctions, classes, fichiers, répertoires, ...).
- la lisibilité des éléments du programmes doit être facilitéeLa plupart des environnements de programmation possèdent des capacités de formattage du code.;
- la "connaissance optimale" nécessite de sélectionner les LS et de construire les programmes selon un principe de **clarté**. Ce principe s'associe à la définition de "chemins" optimaux (confer section 3.4) permettant d'avoir accès aux descriptions d'une caractéristique. Ce critère propose au programmeur :
  - de porter plus spécifiquement ses choix sur des LS standardisés et sur des LS possédant un caractère simplificateur en nombre et en diversité (petit nombre de concepts mis en relation);
  - de modéliser les éléments des programmes en privilégiant leurs descriptions simplificatrices (finalité claire et non complexe).

E - L'informatique complexe

L'Analyse Systémique décrite propose un cadre général pour l'étude des systèmes complexes. Elle se base sur une vision systémique de l'analyseur qui est supposé posséder deux caractéristiques :

- un **Savoir** visant à gérer la connaissance sous la forme de Langages Systémiques (LS) et d'Informations Systémiques (IS);
- un système de raisonnement responsable de l'Analyse Systémique.

Du fait des qualités d'automatisation, de rapidité et de puissance de traitement de l'informatique, l'analyse de systèmes de plus en plus complexes devient possible. Concernant cette informatique, l'enjeu des prochaines années se trouve dans sa capacité à s'**adapter** aux besoins actuels de **maîtrise** de la gestion d'une information de plus en plus nombreuse et complexe. Il en existe d'ailleurs quelques prémisses"Grid Computing", "UML", Services Web, E-science, logiciels dédiés (environnements complexes de modélisation, flux de données, gestion du travail en équipe, ...) , AspectJ, Encyclopédie Wikipedia, Google, "datamining ", ... dans beaucoup de domaines.

Au demeurant, l'informatique se perfectionne continuellement en s'attachant à nous "rendre service" dans nos besoins journaliers. Cela concerne essentiellement:

- le **traitement** de l'**information complexe** (gestion de la connaissance, analyse multi-critères, gestion d'une information multi-disciplinaire, ...);
- l'**intégration** de l'informatique dans nos mécanismes de raisonnement.

Dans le second cas, l'informatique doit proposer des outils d'interface de communication permettant une optimisation de la transmission des informations, par exemple en "épousant" étroitement nos mécanismes de réflexion (ergonomie) et en devenant plus réactif. Ainsi sur Internet, une tendance actuelle de développement se situe dans l'implémentation de "partenaires" (émetteurs et récepteurs d'informations) de type logiciels ou services Web. Ceux-ci **communiquent** entre eux et, pour quelques uns, possèdent un certain **degré d'adaptation** leur permettant d'établir un dialogue avec d'autres "partenaires", qu'ils soient de nature logicielle ou humaine.

Une solution à ces types de problèmes complexes se trouve dans le couplage entre l'informatique et la Systémique. Toutes les deux se complètent merveilleusementOn trouvera par ailleurs quelques informations supplémentaires sur le site suivant : http://systemique.site.voila.fr. en apportant, pour l'une, ses qualités de gestion de l'information (rapidité, automatisation, puissance), et pour l'autre, ses qualités d'analyse de systèmes complexes.

L'Analyse Systémique décrite dans ce document peut servir de base conceptuelle au développement d'une nouvelle discipline, l'informatique complexe ("informatique systémique"), axée sur la réalisation de ces systèmes de traitement de l'information complexe.

F - Référence

[1] J. L. Le Moigne

La modélisation des systèmes complexes

Afcet Systèmes Bordas - 1990 - ISBN: 2-04-019704-4