# Maîtrise de l'entropie des systèmes d'information

■

René Mandel

Résumé

**Les systèmes d'information deviennent de plus en plus complexes, et leur logiciel étendu, imbriqué, protéiforme. L'accroissement de cette entropie[1] a comme origine les interdépendances « naturelles » d'initiatives dispersées, difficiles à coordonner de façon optimale. La thèse exposée ici, est que l'accroissement de l'entropie des SI n'est pas une fatalité et que la complexité croissante résulte, d'une part, de la dérive naturelle des SI et, d'autre part, des microdécisions de construction des logiciels, avec des mécanismes de complexification pervers. En particulier, autour des composants les plus partagés, dits composants de référence, se développent des échanges et interactions contingentes, avec des mécanismes de complication dus aux multiples variantes de modalité et de technique d'interfonctionnement. L'accumulation de particularismes crée des échanges en forme de « plat de spaghetti », ou de « Tour de Babel ». Cet effet cumulatif détériore progressivement l'état des systèmes, par rapport à une organisation parfaite de la population de composants, où les composants de référence seraient utilisés à l'identique, dans tout l'espace du SI, lors des intégrations successives, et dans la durée.**

**Une approche possible pour réduire cette tendance naturelle à la complexité est :**

- **De spécialiser des routines sur les conversions : de syntaxe, de latence, d'interaction. Ces routines limiteraient, lors des intégrations de composants, la complication par combinatoire des particularismes. L'accroissement de l'entropie serait ainsi maîtrisée ;**
- **D'objectiver une architecture, réduite mais durable, qui assemble les composants de référence.**

**Une telle architecture doit être pérenne, puisqu'elle constitue le socle du SI. De ce fait, elle rend le SI flexible, car il peut s'organiser comme un lego autour de l'architecture.**

Mots clés

**Entropie, complexité, architecture logiciel, MDM, flexible, migration, système d'information** ■

## 1. La maîtrise de la complexité des SI

Les systèmes d'information résultent d'une réalité de plus en plus complexe : multiplication des capteurs, extension des domaines applicatifs, empilement des technologies, sédimentation applicative, …On constate qu'au cours du temps les systèmes deviennent de plus en plus complexes, coûteux à maintenir et difficiles à faire évoluer.

La maîtrise de la complexité des systèmes est donc un défi économique. Mais c'est aussi un défi pour les sciences du logiciel. Ce défi peut être vu de différentes façons : ici on s'intéresse plutôt aux lois d'association des composants, au comportement des décideurs des projets et à la « démographie » des composants au sein d'un système d'information.

## 2. La perversité de l'intégration des systèmes d'information

Dans l'activité de production de logiciels, depuis plusieurs décennies, les limitations sont connues. Une des questions clés est celle de l'intégration des composants pour constituer, par assemblage, des « applications » et, de façon plus globale, des systèmes d'information.

### 2.1. L'algèbre des composants du SI

On peut donc avoir une vision globale d'un SI, comme un tout qui évolue :

- par l'assemblage et le désassemblage de composants,
- par les cycles de vie des composants eux-mêmes.

Ce cycle d'un SI, qui n'est pas celui de l'exécution mais celui du logiciel et des composants techniques, est complexe et résulte de toutes ces évolutions.

Chaque composant a son propre cycle de vie, avec une phase de création, des épisodes de maintenance, de refonte, de nécrose (abandon progressif)….

Une forte particularité des SI est l'interdépendance entre les composants. En effet, de multiples interactions sont nécessaires, voulues ou même involontaires.

On peut dire qu'entre deux composants il y a une « fonction » d'interaction, « l'intégration », qui permet

1 - En thermodynamique, grandeur qui caractérise le degré de désordre de la matière

de considérer l'ensemble qu'ils constituent comme un nouveau composant. Ce nouveau composant peut lui-même interagir avec d'autres. Cette fonction est « transitive » et, par assemblages successifs, par associativité, se constituent des ensembles de plus grande taille.

Ainsi, du point de vue ensembliste, les composants obéissent à des lois particulières.

On pourrait faire l'analogie avec la physique moléculaire, et avec le comportement des atomes entre eux [1]. Les composants, en s'intégrant, respectent une loi de composition, une « algèbre », qui leur est propre. On pourrait faire l'hypothèse d'une algèbre des composants « parfaite » respectant une loi de composition associative ((x * y) * z = x * (y * z) et commutative : x*y*z = x*z*y. Mais en pratique, les particularités d'intégration, comme la dispersion des projets entre équipes et dans le temps, laissent penser que cette hypothèse n'est pas réaliste dans l'état de l'art actuel.

### 2.2 Dérive vers la complexité

Au cours du temps, de nouvelles interactions apparaissent, par « transitivité » de l'intégration : de proche en proche des dépendances involontaires et non objectivées se créent ainsi. En effet, l'intégration est maîtrisée « par proximité » (des maîtres d'ouvrage, des maîtres d'œuvre, technique, temporelle…) mais la propagation de la connaissance est toujours limitée et repose sur une base humaine.

Ainsi, au-delà des cercles de proximité, des dépendances **indirectes et involontaires** sont créées. Ces imperfections s'accroissent au cours du temps, car la connaissance se dilue progressivement, et de nouveaux besoins apparaissent auxquels il faut répondre par de nouvelles intégrations. De plus, des particularismes ou modes technologiques introduisent des variantes. Ainsi, dans l'informatique classique, l'intégration est souvent réalisée par échange d'informations. La construction de ces échanges au coup par coup complexifie le système : l'intégration de n composants interagissant avec p composants selon le même échange logique (les mêmes données, indépendamment de leur syntaxe d'échange) tend à multiplier arbitrairement les cas d'intégrations, c'est l'effet spaghetti.

Dans la construction d'un SI plus actuel, on n'est pas à l'abri de cette complexification, par exemple la multiplication des API pouvant générer redondances, recouvrements, variantes entre API. De ce fait, les SI sont « sur-complexes » par ajouts opportunistes et intégrations privilégiant la proximité et le court terme.

### 2.3 L'intégration des données « de référence »

Dans tous les SI on constate qu'il existe le besoin de partager des « données de référence ». Le rôle clé de ces données pour maîtriser le désordre des systèmes, bien au-delà de la stricte sphère des SI, est aussi admis par toutes les organisations.

Il s'agit bien, au final, d'intégration de logiciel, qui peut avoir une dérive perverse comme indiqué ci-dessus. En effet, un composant de référence joue un rôle particulier dans un périmètre (la totalité du SI ou un sous-ensemble) : il est susceptible d'être « intégré » avec tous les autres composants de ce sous-ensemble, et cette intégration est parfaitement transitive : un composant intégrant une telle référence hérite de cette faculté et peut servir de référence à son tour, et à l'identique.

En quelque sorte la loi de composition serait associative :

(x * y) * r = x * (y * r)

Et, si H est la loi d'héritage des propriétés de la référence, on pourrait noter :

H(x*y*r)=H(y*r)=H(r).

En réalité, on constate toujours que l'intégration de ces composants est « **imparfaite** » et donne lieu à des aberrations. En quelque sorte, il y a multiplication des héritages d'un même objet de référence. Comment expliquer les imperfections de cette « algèbre » ?

La connaissance sur les composants de référence connaît des limites, d'autant plus fortes que le système est étendu : des composants, qui ne devraient pas être classés parmi les composants de référence, peuvent, pour des raisons variées (absence de connaissance, volonté de contrôle, volonté de modifier le modèle…), être substitués à un composant de référence au sein d'un sous-ensemble. Outre le défaut de connaissance, il y aussi mille autres bonnes raisons : introduction d'une variante, refus de la généricité, une cinématique d'échange particulière (lots, rythme de mise à jour, messages, invocation de services…).

De fil en aiguille, les liens d'intégration s'étendent, dans le désordre, par proximité, sans respecter la qualité de l'intégration proposée par les composants de référence. Ainsi, un maillage aléatoire des composants se crée, et rend le système de plus en plus complexe.

### 2.4 Divergence versus convergence dans la population des composants

Si nous considérons la démographie des composants, on peut voir le processus comme une histoire où, au cours du temps, les composants évoluent, la population change et les liens d'intégration se font et se défont.

Il y a certes une évolution des périmètres, qui va en général dans le sens d'une extension du système et de sa population. Le système devient ainsi plus complexe. Mais il ne faut pas qu'il devienne au cours du temps « sur-complexe », par augmentation des dysfonctionnements indiqués ci-dessus.

On pourrait, par exemple, proposer un modèle probabiliste pour la fonction d'héritage d'un composant de référence, lors d'une intégration :

- Probabilité de choix du « vrai » composant : p

- Probabilité de choix du clone composant de référence le plus proche : 1-p
- Probabilité de dégrader le composant lors de ce choix en affichant une référence différente : 1-p2

À l'issue de cette intégration, ce composant, s'il est utilisé pour une autre intégration, sera dégradé selon une probabilité 1-p*p2

Ainsi, le taux d'imperfection de l'héritage, lors des intégrations, a un effet cumulatif, accroissant progressivement la sur-complexité. Cette loi aboutit à une dégradation systématique : ce phénomène, s'il se produit avec les composants de référence, aboutit à un système en « crise sanitaire », car ces composants, très partagés, sont agents de propagation de la sur-complexité.

Par contre, si on réussit, au cours du temps, à corriger la loi d'héritage, on aboutit à l'effet inverse : de nouvelles imperfections ne se produisent plus, les anciennes imperfections sont naturellement résorbées au fil des intégrations (le taux de composants issus d'une intégration imparfaite diminuera progressivement au sein de la population). La complexité se réduira naturellement.

### ◗ 2.5 Le vecteur de complication

De fait, il y a plusieurs raisons, dans l'état actuel de la technologie IT, qui expliquent la création de variantes lors des intégrations de composants de référence :

- Absence de modélisation de la subsidiarité : on veut faire porter au composant de référence un concept central unique, qui ne correspond ni au besoin, ni à la réalité connue et perçue par les utilisateurs, les acteurs de l'entreprise, les clients. Ceci amène à créer autant de clones qu'il y a de visions du concept, et donc de définitions et instanciations de l'objet. En outre, ces clones ne sont pas stables au cours du temps, car l'organisation évolue et les affaires se transforment. Des divergences sont en réalité dues à une subsidiarité cachée, qui n'est pas objectivée,
- Pas de gestion des multiples latences dans les échanges (flux à différentes échéances, stock sous forme fichier, accès par API, …),
- Séparation entre mondes (Batch, messages, …) et multiplication des protocoles et formats,

Mauvaise gestion de la datation, de l'historique et du cycle de vie des données.

Ainsi, lors d'une intégration, guidée par les exigences du moment, on applique, à la demande de composant de référence, un vecteur de complication (effet « Tour de Babel). Et cette complication se reproduit à chaque nouvelle intégration.

## 3. Agir sur le cœur de l'intégration

Comme dans d'autres sciences, on peut étudier le système dans sa globalité, ou dans son fonctionnement détaillé. Par exemple, la physique a d'abord dégagé des lois générales, bien que l'intuition de l'existence des atomes soit très ancienne. La physique quantique est venue ensuite formuler les modèles de comportement et d'interaction à l'échelle des atomes.

Mais, en physique, l'homme n'intervient pas dans ces interactions atomiques. Dans le cycle de vie des SI, par contre, les comportements humains, aussi bien ceux des individus, et surtout ceux dus aux conditionnements sociaux, aux enseignements académiques, sont déterminants.

### ◗ 3.1 Rendre le marché des composants plus parfait

Dans les choix d'intégration successifs, le souhait est qu'ils soient optimaux, et qu'ils améliorent le retour sur investissement du système. La question du ROI d'un SI est primordiale : un SI coûte et apporte de la valeur tout le long de sa durée de vie, et on ne peut juger de son utilité que sur cette durée. Nous avons proposé un modèle mathématique pour formuler ce ROI [2].

Les composants de référence trouvent leur justification dans leur intégration aux autres, comme s'il y avait une transaction interne, avec un prix de cession. Cela pose plusieurs questions :

- le « marché » interne des composants doit être « parfait »,
- les composants de référence ne doivent pas être dégradés par des vecteurs de complication,
- il faut avoir investi sur la bonne population de composants de référence.

#### *3.1.1 Agir sur le marché interne*

Le besoin est de rendre, concernant les composants de référence, l'information symétrique, dans **un marché « parfait »**, pour que les « acheteurs » aient une connaissance des avantages, des garanties, des risques.

Cette **transparence** du marché interne des composants de référence, permettrait d'éviter les dérives mentionnées ci-dessus, quand elles sont dues à une connaissance imparfaite du patrimoine.

Cela implique aussi que les composants de référence soient disponibles **« en avance de phase »** par rapport aux cycles de développement des autres composants. Que le développement soit réalisé en Cascade ou selon des méthodes plus agiles, les composants de référence doivent être connus en amont des projets, et, idéalement, accessibles en simulation très tôt : l'impact de ces choix va bien au-delà du SI et concerne la sphère sociale, dont on ne peut décréter l'agilité.

#### *3.1.2 Isoler les vecteurs de complication dans des routines dédiées à la traduction*

Une autre clé pour optimiser les intégrations autour des composants de référence est de doter ceux-ci de capacités d'intégration « génériques » leur permettant d'être assemblés dans tous les contextes, de **façon « parfaite », sans surcoût, et sans complication**. À cet effet les qualités sont :

- L'anticipation d'évolutions futures, pour garantir aux projets un « service » adapté aux nouvelles technologies et aux exigences futures : préfigurer, pour les projets, des **modalités nouvelles**, plus alignées sur les possibilités actuelles et les attentes imposées au SI (tout connecté, mobilité, traçabilité, transparence, toutes sources, agilité,...).
- L'offre d'interfaces variées pour une insertion « **non-intrusive** » au sein du patrimoine existant : admettre toutes les modalités d'échange et d'intégration, pour une cohabitation avec le patrimoine existant sans surcoûts, et dans des **pas de temps courts.**

Cette capacité est obtenue par le développement de services de données utilisant une **bibliothèque d'intégration de données**, et la conception de modèles génériques pour les données de référence.

Une architecture d'interopérabilité peut, par exemple, être constituée pour s'appliquer aux mondes de la répartition de flux (traitement par lots, messages) et à celui de l'orchestration de services (services web, API, …) et les faire converger.

En somme, la création de ces composants traducteurs, routines s'interposant dans les échanges et interactions, évite l'effet spaghetti ou Tour de Babel,

### ◗ 3.2 Investir sur les bons composants de référence

On comprend bien que tous les efforts décrits ci-dessus sont vains si le choix des composants de référence s'avère inadéquat. Cette question est cruciale, puisque on voit que les composants de référence structurent le SI, lors d'un projet, mais surtout dans la durée. En outre, les composants de référence ne sont pas indépendants les uns de autres. Ils constituent un ensemble synergique. Il est convenu d'appeler cet ensemble une « architecture », par analogie avec celle des bâtiments.

## 4. Définir une architecture pérenne pour un SI flexible

L'assemblage des composants au sein d'une architecture est au cœur de la problématique de maîtrise de la complexité. Schématiquement, on peut définir une architecture comme un classement des composants en deux sous-ensembles :

- Les composants C1 constituant l'architecture elle-même, « composants d'architecture » qui donnent une capacité d'architecture, une fonction d'architecture,
- Les composants C2 qui sont associés par cette architecture, mais n'interviennent pas dans la fonction.

Cette définition ensembliste peut s'appliquer en plusieurs niveaux, un ensemble de composants C2 pouvant à son tour être classé en fonction d'une sous-architecture C2.1 et C2.2.

L'architecture pose donc deux questions majeures :

- Quel classement des composants ?
- Quel niveau de granularité ?

Sans entrer dans les détails, on peut voir cela selon trois classements des composants :

- Le taux de partage
- La stabilité versus la volatilité
- La simplicité versus la complexité

### ◗ 4.1 Taux de partage des composants

Si, avant de concevoir une architecture, on pouvait, pour chaque composant, noter tous les liens logiques qu'il a avec d'autres composants, on verrait une courbe de concentration, où un petit nombre de composants seraient impliqués dans une grande proportion des liens.

Ces composants, naturellement au centre de la collectivité, ont vocation à devenir composants de référence, avec l'avantage d'un effet de levier (leur évolution concerne beaucoup d'autres composants) et l'inconvénient d'effets domino (leur dégradation a un impact négatif fort, voire un impact systémique fatal).

À l'opposé, un grand nombre de composants sont situés en périphérie et n'ont pas d'effet de levier.

### ◗ 4.2 Stabilité versus volatilité

Un autre classement concerne la stabilité du modèle d'un composant. Certains composants sont pratiquement invariants, ou évoluent plus lentement que les autres. À l'opposé, des composants sont fréquemment remis en cause dans leur modèle même.

### ◗ 4.3 Simplicité versus complexité

Enfin certains composants sont simples à appréhender, alors que d'autres apparaissent complexes, avec de nombreux détails que l'analyse devrait expliciter.

### ◗ 4.4 L'architecture flexible

En combinant les classements énoncés ci-dessus, on peut aboutir à une architecture idéale dans la durée : Les composants d'architecture sont fortement partagés, stables dans leurs modèles (invariants) et relativement simples. L'architecture sera donc durable. L'ensemble, ainsi architecturé, sera flexible, car les autres composants seront :

- Évolutifs, mais ne remettant pas en cause dans leur évolution, l'architecture, donc sans impact fort sur le reste de la population,
- Interchangeables,
- Vus comme des boîtes noires de complexité.

En somme, la mise en œuvre d'une « Architecture Flexible » est probablement la meilleure garantie pour limiter, dans la durée, l'accroissement de la complexité. Voir la référence [3] à ce sujet

## Références

[1] *http://www.value-architecture.com/2015/03/les-limites-de-la-complexite-des-si.html*

[2] René Mandel : Systèmes d'information : modélisation du ROI de projets d'infrastructure ; Génie Logiciel, n° 71, pp. 40-47, décembre 2004

[3] *http://www.value-architecture.com/2016/02/larchitecture-flexible-en-pratique.html*

## Biographie

**René Mandel.** Né en septembre 1942 à Saint-Étienne, fils de Jean Mandel (Polytechnicien, X26, Corps des Mines, enseignant-chercheur en Mécanique), René Mandel fait ses études à Paris et entre à Polytechnique à 19 ans. ENSAE 1966, il intègre le corps des Administrateurs de l'Insee. À l'Insee il est le premier chef de la Division Logiciels, et fait développer des logiciels statistiques pionniers (avant SAS), et en particulier Leda, largement diffusé avant 2000. Fondateur de la société ORESYS, il est un des acteurs historiques de l'urbanisme des systèmes d'information en France, et initiateur du club URBA-EA. Il est l'auteur de l'ouvrage « De la stratégie business aux systèmes d'information » (Hermès 2006) et de nombreux articles professionnels. Il est l'inventeur des concepts de « trame business », de « puits de données » et d'« architecture flexible ».

Blog :*http://www.value-architecture.com/*

Profil LinkedIn : *http://fr.linkedin.com/in/renemandel/*