Modélisation par le flux

Milovann Yanatchkov[1,1]

[1]UMR MAP-MAAC 3495 CNRS/MCC, ENSA Paris La Villette, 144 avenue de Flandre, 75019 Paris, France

Résumé. Nous présentons le principe de modélisation par le flux comme un paradigme de modélisation distribuée adaptée à la phase initiale de conception de l'architecture s'inspirant du principe de la programmation par le flux. Au moyen d'un serveur 3D, un nombre arbitraire de logiciels peuvent coopérer en temps réel à la création d'un « modèle collaborant » conçu comme la somme des opérations de modélisation rassemblées au-delà des strictes limites de chacune des applications participantes.

Mots-clés. Conception numérique, Modélisation distribuée et procédurale, BIM.

Abstract. We present the paradigm of flow-based modeling as a form of distributed modeling tailored for the initial phase of design inspired by the concept of flow-based programming. By means of a 3D server, an arbitrary number of applications can cooperate in real-time to produce a "collaborating model", which exists as the sum of the whole modeling operations collected beyond the limits of each of the software involved.

Keywords. Digital design. Distributed and procedural modeling. BIM.

1. Introduction

Les outils numériques ont profondément renouvelé les modes de conception de l'architecture. Désormais, les éléments descriptifs du bâtiment sont stockés et véhiculés sous forme numérique formant potentiellement un *continuum* entre les données issues de la phase initiale de conception et les documents marquant l'achèvement de l'objet construit, pilotant parfois les machines de fabrication.

Depuis leur apparition, ces outils ont connu deux évolutions importantes. La maquette numérique (BIM), d'une part, est en train de bouleverser les fondements mêmes de la discipline en ce qu'elle fait évoluer les méthodes traditionnelles de conception et de représentation (élaboration des plans, coupes et façades). La démocratisation des langages de programmation visuelle, d'autre part, représente un changement important de paradigme

[1] my@rvba.fr

en ouvrant les méthodes de conception aux logiques procédurales et computationnelles du monde numérique.

1.1 Modélisation orientée objet

Dans le courant des années 1980 une première vague de logiciels de Conception Assistée par Ordinateur (CAO) fournit des outils de dessin numérique qui restent proches des instruments traditionnels sous la forme d'une sorte de planche à dessin numérique reproduisant les gestes de la géométrie descriptive à travers la manipulation d'éléments tels que lignes et arcs. Pendant cette période et durant une vingtaine d'années, la baisse des coûts et l'augmentation de la puissance des ordinateurs personnels (PC) vont démocratiser ces usages jusqu'au tournant des années 2000 qui signe le basculement définitif de la profession vers le numérique.

Pendant que se démocratisent les logiciels de CAO, un nouveau paradigme s'élabore progressivement autour de l'idée de maquette numérique. Il ne s'agit plus de dessiner des représentations planaires de l'architecture mais d'assembler des éléments prédéfinis tels que murs, portes ou menuiseries, puis d'effectuer des opérations de section et de projection du modèle pour en générer automatiquement des épures. Cette méthode présente l'avantage de maintenir une cohérence parfaite entre l'ensemble des représentations et prévient l'apparition d'incohérences au cours de l'élaboration du projet. Le fait de manipuler des objets prédéfinis représente un gain notable de productivité par rapport à la CAO et puisqu'il s'agit de manipuler des entités prédéfinies, nous pouvons parler à leurs propos de modélisation orientée objet (Howell et Batcheler, 2005).

1.2 Modélisation procédurale

Dans le courant de la décennie 2000-2010, l'intégration de langages de programmation visuelle aux instruments de modélisation 3D contribue à faire évoluer notablement les modalités de conception. Ces instruments permettent de manipuler les données géométriques du projet par l'intermédiaire d'opérateurs qui, reliés entre eux par des points d'entrées et des points de sorties, forment une arborescence décrivant visuellement des algorithmes génératifs. Grâce à leurs interfaces visuelles, ces outils sont facilement manipulables par des non-spécialistes et permettent à un utilisateur sans connaissance préalable des langages formels d'élaborer de véritables systèmes computationnels. Malgré tout, cette approche présente de sérieuses limites au-delà d'un certain seuil de complexité. Pour les concepteurs les plus avancés, l'écriture directe de codes sous formes textuelles reste la solution la plus adaptée.

1.3 Modélisation par phase

Depuis 2010, l'évolution de ces instruments reste concentrée au niveau du projet d'architecture dans sa phase avancée de conception en rapprochant toujours plus les représentations numériques des réalités constructives et des standards industriels. Or, la conception architecturale est un processus ayant pour origine une problématique abstraite, puis pour finalité un objet concret, généralement à l'issue d'un processus menant du moins composé vers le plus composé, du moins ordonné vers le plus ordonné. Cette phase amont de la conception reste un sujet peu abordé par les acteurs de l'industrie du logiciel, peut-être en raison de son caractère non spécialisé, mal défini, voire informel. Nous proposons donc d'analyser les instruments numériques en distinguant deux phases bien déterminées : une phase initiale de prise de forme, caractérisée par un travail sur des hypothèses dont les

présupposés peuvent fortement varier et donner lieu à de fréquentes itérations du processus de définitions à différentes échelles, impliquant de grandes amplitudes dans les modifications des rapports entre le tout et la partie ; une phase avancée de production marquée par la stabilisation croissante de l'objet de la conception et par un travail d'ajustement des détails dans la représentation concrète de l'édifice à construire. En nous référant à ce processus graduel de conception de l'architecture, nous analysons à présent la nature des instruments numériques de conception en distinguant derrière l'idée unique de modèle 3D, une diversité de modèles (géométriques, computationnels, informationnels).

2. Modèles pour la conception numérique

2.1 Modèles géométriques

Le projet d'architecture est en première instance une simple idée ou une pure intention, une abstraction hors de toute réalité matérielle. Le passage de l'idée abstraite à sa première expression spatiale s'opère par l'intermédiaire de schémas, diagrammes ou croquis. Ensuite le projet atteint progressivement un niveau de définition plus élevé en vue de sa construction et nécessite l'élaboration d'un ensemble important de documents. Ce passage de l'abstraction à la concrétisation de la représentation de l'architecture s'exprime clairement dans les usages d'*outils génériques* et d'*outils métiers*. En phase initiale, il est courant de travailler avec des logiciels qui ne sont pas directement issus du monde de l'architecture mais plutôt du monde de l'image animée ou du jeu vidéo : l'expressivité y prime sur la précision, l'image compte autant que l'épure. En revanche, en phase avancée de production, la multiplication et la complexification des documents de conception rendent nécessaire le recours aux instruments CAO et BIM qui ont été conçus spécialement en termes de gains de productivité (Autodesk, 2002).

En matière de géométrie, la distinction entre *outils génériques* et *outils métiers* se vérifie par l'emploi de modèles surfaciques polygonaux pour les premiers et de modèles hautement composés ou volumiques pour les seconds. Les modèles surfaciques sont simplement constitués de points reliés entre eux et formant des surfaces privées d'épaisseur. Ils sont adaptés au niveau de la maniabilité pour la phase initiale de conception, et sur le plan de la légèreté quand il s'agit de répondre à des impératifs de performance d'affichage. Les modèles volumiques quant à eux, sont basés sur des représentations géométriques par les contours, et sont hiérarchiquement composés (solides, surfaces, contours, faces, arêtes, points). Ces modèles offrent moins de liberté en matière de maniabilité mais répondent mieux aux problématiques propres à l'architecte ou à l'ingénieur, notamment quand il s'agit d'effectuer des opérations complexes sur la géométrie, telles que des opérations booléennes permettant de produire des sections sur le modèle.

2.2 Modèles computationnels

En phase initiale, l'architecte dispose en général d'un programme bien défini et d'un ensemble de données et de contraintes précises sans que cela ne forme nécessairement un ensemble cohérent. À partir de ce champ ouvert et peu structuré, la genèse du projet peut alors se concevoir comme le processus itératif des choix successifs de définition et de composition. En informatique on peut assimiler ce processus à l'enchaînement des procédures qui s'appliquent aux données du modèle 3D. Nous pouvons classer les instruments computationnels selon leur adéquation à la phase initiale de prise de forme ou à

la phase avancée de production en distinguant deux types d'approches (paramétrique et algorithmique) et trois types d'instruments (modèles objets, langages de programmation visuels, langages de programmation textuels).

Les instruments BIM sont basés sur des modèles géométriques hautement composés (BREP) ayant des aspects de modélisation orientée objet. Ces objets sont organisés selon une structure spécifique au bâtiment (niveaux, structures, distribution, etc.) qui prédétermine et contraint très fortement le cadre spatial, relationnel et signifiant dans lequel peut se dérouler la conception. Dans ce cadre fixe, un ensemble d'objets prédéfinis (portes, murs, dalles, etc.) peuvent être assemblés selon des logiques et des contraintes spatiales spécifiques. Cette approche de la modélisation paramétrée se présente donc comme un domaine contraint contenant des objets prédéfinis qui convient principalement à la phase avancée de production.

Les langages de programmation visuels représentent un niveau plus avancé de « programmabilité ». Ils étendent les fonctionnalités existantes de l'instrument hôte dans lequel ils sont embarqués et peuvent donner accès à des fonctionnalités de génération géométrique algorithmiques à travers la combinaison d'opérations sur des primitives géométriques. Ils sont plus adaptés à la phase initiale de conception, car ils opèrent sur un champ ouvert et moins contraint. De même, les langages de programmation textuels offrent encore un degré supplémentaire de liberté, car ils opèrent sur une des représentations formelles de plus bas niveau. Nous pouvons par analogie, et en gommant les détails de fonctionnement de ces instruments, comparer les premiers au principe de la programmation orientée objet et les seconds au principe de la programmation procédurale.

2.3 Modèles informationnels

En phase initiale de conception, l'élaboration de représentations spatiales est largement ouverte à toutes sortes de libertés expressives, voire à des formes contradictoires ou à des géométries impossibles. En revanche, en phase avancée de production, les documents graphiques sont soumis à des règles très spécifiques et bien codifiées. D'un point de vue numérique nous pouvons également faire cette distinction au niveau des formats de données et des structures informationnelles.

Dans le cadre de l'interopérabilité et de la consistance des opérations de modélisation des données du bâtiment (BIM), le format IFC (*Industry Foundation Classes*) s'impose comme un standard adopté par la majorité des acteurs du BTP. Extension du langage de modélisation industriel STEP, il est hautement structuré en fonction des applications métiers et des produits manufacturés qu'il décrit (Lee, 1999) et donc adapté à la phase avancée de production. En revanche la phase initiale de conception numérique n'ayant pas fait l'objet d'un travail normatif d'interopérabilité, il n'existe pas de standards bien définis. Nous pouvons cependant émettre l'hypothèse que les formats de modèles surfaciques les plus élémentaires (proches du format *OpenGL*) sont les plus adaptés à cette phase de conception. Encore une fois nous pouvons tenter de distinguer ces deux approches en remarquant que le format IFC entre le cadre des *modèles de données* et que le format *OpenGL* est lui une *interface de programmation applicative* (API).

2.4 Polarisation des phases de conception numérique

Au terme de cette analyse, nous voyons nettement se former une opposition entre les modèles, les instruments et les formats suivant leur adéquation à l'une ou à l'autre des phases de conception. D'un côté, nous avons la phase initiale, un domaine de recherche où les instruments génériques basés sur des géométries surfaciques sont plus adaptés pour

manipuler des primitives géométriques et des objets faiblement composés (*OpenGL*). De l'autre, nous avons la phase avancée, un domaine concentré sur la production du bâtiment où les instruments métiers basés sur des géométries solides sont plus adaptés pour manipuler des objets architecturaux fortement composés (IFC). Nous pouvons classer ces modèles suivant leur adéquation à l'une des phases conception (tableau 1).

Tableau 1. Classement des modèles suivant la phase.

Modèle / Phase	Phase initiale	Phase avancée
Géométrique	Géométrie polygonale	Géométrie solide
Computationnel	Programmation procédurale	Programmation objet
Informationnel	API	Modèle de données

Nous pouvons décrire cette séparation des domaines en termes de polarisation d'un point de vue numérique entre une phase originelle que nous qualifions de *bas niveau* (faiblement composée et peu définie) et une phase avancée que nous qualifions de *haut niveau* (fortement composée et très définie).

Les instruments de type BIM qui correspondent à la phase avancée de conception et au domaine de *haut niveau* sont également des vecteurs d'interopérabilité des données de modélisation du bâtiment, mais nous remarquons que paradoxalement, ce rôle contribue à une forte verticalité de l'intégration des technologies par éditeurs et au cloisonnement entre applications. C'est en réponse à cette problématique que nous proposons le principe de *modélisation par le flux* comme un paradigme de modélisation plus adapté à la phase initiale car plus ouvert et plus horizontal. En prenant appui sur la potentialité des réseaux informatiques nous cherchons à mettre en concordance des environnements hétérogènes et à décloisonner le cadre dans lequel s'opère la conception numérique.

3. Principe de modélisation par le flux

3.1 Modèles distribués

La généralisation du travail en réseau par l'intermédiaire de serveurs de fichiers et le développement de l'informatique dans les nuages – stockage à distance dans des centres de données – représentent une troisième évolution importante des usages numériques de conception (Afsari *et al.*, 2016). La distribution des opérations de conception à travers un réseau d'opérateurs présents parfois sur des sites de production distants et la diversification croissante des applications disponibles donnent au contexte actuel le caractère d'une forme d'écosystème numérique.

En génie logiciel, on parle d'architecture distribuée lorsqu'un système n'est plus pensé de façon monolithique mais plutôt comme une pluralité d'éléments indépendants les uns des autres échangeant des informations à travers des canaux de communication (Arpaci-Dusseau et Arpaci-Dusseau, 2015). Le principe porté par le BIM d'une maquette numérique distribuée et décentralisée entre dans le cadre de cette conception.

Nous pouvons distinguer deux approches BIM de distribution des opérations de conception. Dans le premier cas, nous trouvons des solutions fonctionnant à partir d'une famille d'applications appartenant à un même éditeur et fonctionnant sur des formats de données natifs ou bien restreints à des partenariats spécifiques d'interopérabilité. Cette approche fait l'objet d'une vive concurrence entre les grands acteurs de l'industrie logicielle et ne garantit pas une parfaite neutralité et une bonne interopérabilité. Une autre approche

consiste à développer des applications spécifiques telles que des bases de données au format IFC. Ces deux solutions sont adaptées à la phase avancée de production en ce qu'elles sont basées sur des outils métiers et sur le format IFC.

Une architecture logicielle de type client/serveur, basée sur des formats de données plus génériques que celui de l'IFC semble être une solution plus adaptée à la phase initiale de conception. Quelques initiatives d'interconnexions logicielles ont déjà été amorcées par des acteurs extérieurs aux applications métiers. Certaines de ces solutions sont propriétaires et plutôt de *haut niveau*[2]. D'autres sont développées sur un modèle *open source* à partir de technologies orientées web[3]. Nous avons développé le projet Echo[4] en nous basant sur *Verse*, un serveur de données purement dédié à l'échange de données 3D. *Verse* a été développé par Eskil Steenberg et Emil Brink pour l'établissement suédois *KTH Royal Institute of Technology* (Brink *et al.*, 2006). Il s'agit d'un protocole et d'un serveur permettant d'échanger des données 3D en temps réel, initialement conçu pour la création de mondes virtuels. Ses auteurs l'ont également imaginé comme une plate-forme d'interconnexion logicielle. Dans le cadre de cette recherche, nous avons développé des clients (*plugins*) pour les logiciels *Revit*, *Grasshopper*, *Blender* et *Lud*[5] grâce auxquels ils peuvent partager leurs modèles à partir de systèmes *Windows* et *Linux* présents sur des réseaux locaux ou distants.

Le rôle du serveur se limite à faire coexister des objets 3D à travers l'ensemble des applications connectées. Grâce à une instance du serveur active sur le réseau, les applications peuvent se connecter par l'intermédiaire de leurs clients et déposer leurs données ou souscrire à des données déjà présentes sur le serveur. Pour chaque modification effectuée sur l'une de ces données, le serveur diffuse (*broadcast*) et répercute ces nouvelles informations sur l'ensemble des applications participantes. Le principe de la mise en commun des données 3D étant acquise, la question se porte à présent sur la nature de ces données. Nous avons précédemment posé le principe d'un domaine de *bas niveau* pour la phase initiale et identifié la géométrie polygonale et le format *OpenGL* comme étant les plus adaptés. Le protocole *Verse* répond à ces prérequis en ce qu'il définit un format 3D polygonal réduit à l'essentiel. Il s'agit donc de partager et de faire transiter des informations *par le bas*, et le rôle de chaque client est de *réduire* la représentation géométrique interne de l'application participante à son expression la plus minimale : sommets, triangles et quadrilatères.

3.2 Modèles collaborant

Les canaux de communication entre applications étant désormais actifs grâce à l'action d'un serveur central, nous pouvons à présent envisager les possibilités que nous offre ce système en matière d'usage. La première application est évidemment de pouvoir rassembler différents acteurs dans un espace commun afin d'échanger sans contrainte des données et de faire intervenir plusieurs opérateurs sur un même espace ou sur un même objet. À ce stade, l'usage de la plate-forme reste très proche de ce que proposent les modèles distribués de *haut niveau* que nous avons évoqués précédemment. Nous trouvons des usages plus profitables à la phase initiale de conception à partir du moment où nous cherchons à composer les éléments présents sur le serveur. Nous pouvons par exemple chercher à fabriquer un objet composite qui mobilise des ressources disponibles à travers les

2 www.konstru.com
3 www.speckle.works
4 www.github.com/rvba/echo
5 www.github.com/rvba/lud

différentes applications participantes. C'est ce que nous avons expérimenté en nous appuyant sur les moyens d'automatisation disponibles au sein des logiciels : objets dynamiques et générateurs de géométrie, interfaces de programmation visuelle et API. À partir de ces différentes ressources nous avons pu élaborer un objet composite résultant d'une chaîne d'opérations de modélisation distribuées en appliquant le principe suivant : chacune des applications participantes récupère un objet bien identifié présent sur le serveur représentant un état intermédiaire de la modélisation. À partir de cet état E bien déterminé, l'application peut modifier ou adjoindre de nouveaux éléments sur tout ou partie de l'objet avant de déposer sur le serveur un nouvel état de l'objet E+1. Il se constitue ainsi une suite d'opérations régies par un principe d'antériorité où nous trouvons, en début de chaîne, un objet contenant les données initiales de la modélisation et, en fin de chaîne, un objet combinant l'ensemble des apports de modélisation successifs. Ces différents états-objets étant liés les uns aux autres dynamiquement grâce au serveur, chaque modification d'un état donné entraine la modification de l'ensemble des états successifs jusqu'à l'objet en bout de chaîne dans son état *achevé* (figure 1). Nous déployons ainsi à travers un réseau d'applications hétérogènes le fonctionnement d'une *pile de générateurs*, un principe qui reste généralement inclus dans les strictes limites d'une même application. L'existence de ce modèle distribué porte le principe d'interopérabilité au-delà de l'aspect statique contenu dans l'idée de format et de fichier, pour le porter vers un fonctionnement dynamique et paramétrique. La composition s'opérant en temps réel (dans la mémoire vive de la machine), nous pouvons considérer que les maillons de cette chaîne d'opérations et d'états successifs forment une sorte de modèle virtuel existant en pointillé à travers l'ensemble des applications participantes. À l'image du béton d'un plancher collaborant qui travaille avec le métal lui servant de support, le modèle collaborant fait travailler chacune des applications d'une façon solidaire. Ce modèle collaborant étant la somme des potentialités de modélisation que fournit chacune des applications participantes, il existe en tant qu'entité singulière qui possède des caractéristiques qui lui sont propres et en quelque sorte prolonge et dépasse les possibilités offertes par chacune des applications prises individuellement.

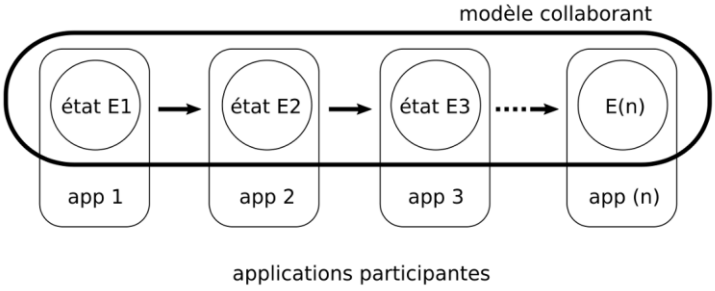

Figure 1. Principe de modèle collaborant.

3.3 Abstraction logicielle

La « composabilité » des applications et l'émergence d'un modèle virtuel *collaborant* inter-logiciels a pour conséquence de redéfinir la place et la nature des logiciels de conception dans le cadre de la phase initiale. En extrapolant le résultat de ces expériences nous identifions au moins deux conséquences possibles. Tout d'abord nous voyons qu'en hébergeant dans un espace *neutre* les données issues d'une variété hétérogène d'applications, le serveur agit comme une interface d'abstraction qui permet de masquer la

réalité des opérations et des calculs opérés par les applications au profit du seul résultat. Pour l'obtention d'un même résultat nous pouvons donc substituer l'une ou l'autre des applications participantes : elles deviennent en quelque sorte interchangeable. Nous voyons dans cette généricité la possibilité de s'extraire des applications nominativement dans une sorte d'*émersion logicielle*. Une autre conséquence que nous pouvons envisager avec ce système est la transformation de l'outil en une ressource. En s'adaptant à l'interface d'abstraction, l'outil devient *mobilisable* comme une ressource computationnelle – lorsqu'il s'agit d'effectuer des calculs sur un modèle – ou opérative – lorsqu'il s'agit de manipuler un modèle. L'instrument passe du statut d'outil manipulé par un opérateur à celui de bibliothèque logicielle employée dans la fabrication d'un système plus vaste. L'outil se transforme en une ressource pour fabriquer de nouveaux outils.

3.4 Modélisation par le flux

Nous proposons l'idée de *modélisation par le flux* comme un paradigme de modèle collaborant et de *logiciel comme ressource* en faisant référence au principe de *programmation par le flux* (*flow-based programming*) de (Morrison, 2013) que nous adaptons au domaine de la modélisation 3D. La programmation par le flux est une approche du développement logiciel qui s'appuie sur la logique du flux de données que nous trouvons dans les langages de programmation visuels. Ceux-ci ont pour origine des problématiques liées au calcul distribué visant à augmenter la puissance globale de computation en décomposant un système en une série d'objets autonomes. Pour résoudre les problèmes de synchronisation que pose cette architecture, le paradigme du graphe s'est avéré être une réponse adaptée (Johnston *et al.*, 2004). La *programmation par le flux* propose de considérer le développement d'une application au niveau du flux des données sur lequel elle opère et non plus sur le développement des opérateurs qui effectuent le calcul. Ce flux est matérialisé par le graphe des connexions qui relient un ensemble d'opérateurs indépendants envisagés comme autant de boîtes noires. Le principe de la *boîte noire* consiste à enclore et séparer strictement l'espace interne d'un calcul du monde extérieur par l'intermédiaire d'interfaces d'entrées et de sorties. Une application développée selon ce principe se *réduit* donc aux seules connexions formant le graphe, reléguant ainsi les détails d'implémentation effective du traitement des données au niveau de chacune des boîtes noires qui peuvent être développées indépendamment de celui-ci. Nous proposons le paradigme de *modélisation par le flux* comme l'adaptation de ce principe à la modélisation 3D en considérant les logiciels participants comme des boîtes noires, et le modèle collaborant comme le graphe de connexion. Le grand mérite de la *programmation par le flux*, selon son auteur, réside dans la pérennité des systèmes qui sont élaborés selon ce principe puisqu'il offre la possibilité de remplacer pièce par pièce les engrenages du système sans porter atteinte à son fonctionnement global. À l'instar des interpréteurs systèmes (*shells*) qui permettent d'élaborer des programmes par assemblage d'autres programmes, la *programmation par le flux*, et par extension la *modélisation par le flux*, portent en eux ce principe d'assemblage et de prototypage. Un principe qui nous offre en quelque sorte la possibilité de créer des prototypes d'instruments et des modèles hybrides à la manière d'un architecte qui élabore ses maquettes d'étude en assemblant librement des éléments de diverses natures.

4. Conclusion

En nous intéressant à la phase initiale de conception numérique de l'architecture, nous avons mis en lumière un domaine de modélisation qui trouve un intérêt dans le principe d'une communication *par le bas* de ses modèles, restreints à une pure géométrie polygonale.

Dans le cas de la *réduction* d'un modèle de *haut niveau* d'un logiciel métier de type *Revit* vers une plateforme de *bas niveau* telle qu'*Echo*, la structuration verticale existante entre l'application et le medium d'échange pourrait faire l'objet d'une comparaison approfondie avec le modèle OSI[6] (*Open Systems Interconnection*) et son architecture par couche, mais elle dépasse le cadre purement expérimental de cette recherche. De même il serait intéressant de considérer, à l'aune de cette communication *par le bas*, les problèmes d'*interopérabilité par le haut* des applications BIM en imaginant par exemple des paliers intermédiaires sur le même principe que le modèle OSI. Les expérimentations que nous avons menées, en faisant communiquer des domaines de *haut* et de *bas* niveau, nous ont conduit à élaborer des objets faisant intervenir des modèles de natures géométriques hétérogènes (modèles polygonaux, paramétrés et volumiques), des interfaces de natures hétérogènes (interfaces graphiques et interfaces textuelles) et des opérateurs de natures hétérogènes (algorithmiques, procéduraux, manuels) sur des systèmes hétérogènes (*Windows* et *Linux*). La fluidification des opérations numériques de conception que nous recherchions a donc été atteinte, et le principe de réduction *par le bas* ainsi mis en évidence pourrait s'annoncer utile, dans le sens inverse, comme un préalable à une recomposition *vers le haut* entre les applications métier.

Bibliographie

Afsari, K., Eastman, C. M., & Shelden, D. R. (2016). Data Transmission Opportunities for Collaborative Cloud-Based Building Information Modeling. In *XX Congreso de la Sociedad Iberoamericana de Gráfica Digital, Blucher Design Proceedings, 3 (1)*, 907-913, Sao Paulo : Blucher.

Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2015). *Operating Systems: Three Easy Pieces.* Arpaci-Dusseau Books.

Autodesk, B. I. S. (2002). *Building information modeling*. White Paper, San Rafael, CA. Autodesk Inc.

Brink, E., Steenberg, E., & Svensson, G. (2006). The Verse Networked 3D Graphics Platform (Part 1). In *Linköping Electronic Conference Proceedings. SIGRAD Conference* (pp. 44-48). Linköping, Sweden: Linköping University Electronic Press.

Howell, I., & Batcheler, B. (2005). Building Information Modeling Two Years Later–Huge Potential, Some Success and Several Limitations. *The Laiserin Letter, 24*.

Johnston, W. M., Hanna, J. R., & Millar, R. J. (2004). Advances in Dataflow Programming Languages. New York, NY: ACM Computing Surveys (CSUR), 36 (1), 1-34.

Morrison, J. Paul (2013). *Flow-Based Programming*. Application Developers' News (1). London.

Lee, Y. T. (1999). Information Modeling from Design to Implementation. In S. Nahavandi & M. Saadat (Eds.), *Proceedings of the Second World Manufacturing Congress* (pp. 315-321). Millet, Alberta, Canada: International Computer Science Conventions.

6 https://fr.wikipedia.org/wiki/Modèle_OSI