Simulation de systèmes

Logiciel de simulation

Logiciel versus langage

La mise en oeuvre d'un modèle est un travail d'envergure. Le choix des outils est crucial. Le plus souvent, c'est un amalgame de composants logiciels

Il est fréquent de devoir choisir entre des logiciel ou la programmation dans un langage générique (C, Java, Python, etc)

Acheter ou construire?

Le choix dépend de plusieurs paramètres:

- coût d'achat vs développement
- possibilité de redistribution et redevance
- pérennité
- gain en temps vs courbe d'apprentissage
- réutilisabilité
- flexibilité
- ressources disponibles et expertise
- contraintes clients et légales

Logiciel commercial

Plusieurs choix: Arena, Flexim, Simio, Autodesk, etc. Voir Google: "simulation software"

Chacun est spécialisé pour une gamme spécifique de problèmes

Coût d'entrée typique de 2500$/an minimum

Coût de formation élevé

Langage de programmation

Maximum de flexibilité, de réutilisabilité, etc

Peut entrainer des côuts de développement important

Viable seulement si on dispose de développeurs compétents avec une méthodologie efficace et mature

Risques et bénéfices

Développer ses propres composants logiciels pose des risques importants qu'il faut gérer adéquatement

Cependant, ça offre la possibilité d'aller au-delà des outils commerciaux. On peut cibler des besoins précis, les traiter avec des méthodes originales et créer un avantage compétitif

Exemple - Calcul de contraintes

On désire connaître les contraintes mécaniques dans un assemblage de pièces en aluminium

C'est un problème standard bien couvert par les outils CAD commerciaux. A priori, il n'y a pas d'intérêt à refaire un logiciel

Exemple - Calcul de contraintes bis

On désire connaître les contraintes mécaniques sur la coque d'un sous-marin stealth

Aucun outil CAD commercial ne permet de calculer avec précision l'interaction entre la coque mince, l'air pressurisée, l'eau. C'est un problème de R&D avec développement de logiciel

Exemple - Flux de matériel (FdM)

On s'intéresse au flux de matières premières dans un procédé manufacturier complexe

Ce problème est fortement dépendant des processus manufacturier et des questions de l’ étude

Le choix logiciel vs programmation est difficile. Il faut évaluer le rapport risque sur bénéfice au cas par cas

Exemple - FdM - Cas A

Hypothèse: on veut développer une expertise et on utilisera des méthodes standards, le simulateur sera un outil de demo général

Il semble alors acceptable d'utiliser un logiciel commercial

Corrolaire: prévoir les coûts des licences et de formation

Exemple - FdM - Cas B

Hypothèse: on veut développer une expertise et on utilisera des méthodes originales

Il peut être acceptable de programmer, surtout si on vise à l'exploiter en conception

Corrolaire: prévoir les ressources compétentes pour le développement et avoir une stratégie à long terme pour la rentabilité, réutilisation, pérennité, etc

Prototype

Il est courant de développer des simulateurs via une approche par prototypes. On utilise alors des outils de développement rapide et une approche agile

Par exemple, si on cible un simulateur en C++, le premier proto pourrait être en Excel, les suivants en Python, pour finalement migrer vers le simulateur finale en Python et C

Autres considérations

Dans le choix logiciel vs programmation, il faut aussi prendre en compte:

- l'interopérabilité avec d'autres systèmes
- les besoins de post-traitement des résultats (animation, statistiques, etc)
- la qualité de la documentation et de la formation disponible
- les besoins futurs (pérennité)
- les avantages compétitifs (se démarquer)

Interopérabilité

Cet aspect est crucial! Dans un contexte réel, le simulateur doit interagir avec les systèmes en place (via sa frontière): BD, senseurs, systèmes de contrôles de procédé, systèmes de gestion de la production, etc

De plus, même si c'est pour une étude (environnement isolé), un simulateur est souvent convertie en outil de production!

Interopérabilité - suite

Un outil préconstruit (ex: Arena) peut être efficace pour une étude, mais est inapproprié pour une conversion en outil de production

En effet, ce genre de logiciel n'est pas prévu pour un déploiement en usine. L'interopérabilité est trop limité, mais aussi il est trop lourd, trop chère, pas suffisamment flexible, pas suffisamment robuste, etc