Digital Engineering

INGENIERIE SYSTÈME & LOGICIELLE

FEUILLE DE ROUTE 2022

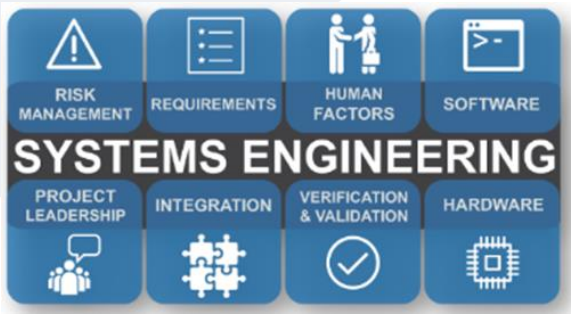
"Vers des méthodes d'ingénierie système et logiciel augmentés et durables pour faire face en confiance à la complexité des systèmes"

L'augmentation de la complexité des systèmes et des logiciels est au cœur de nombreuses préoccupations des concepteurs. Face à cela, le rythme incrémental des innovations en matière de méthodes et d'outils dans le domaine de l'ingénierie des systèmes numériques ne suffit plus pour répondre aux attentes et relever les défis qu'ils soulèvent liés à leur complexité. Les innovations dans le domaine des technologies numériques sont pourtant porteuses de promesses capables d'introduire les ruptures nécessaires pour changer cette situation. Ces innovations concernent notamment l'IA appliquée à l'ingénierie système et logiciel, les avancées dans le domaine des technologies informatiques dites « formelles » (au sens des mathématiques) et les progrès en terme d'environnement de travail collaboratif, une des clefs du succès de l'intelligence collective.

Introduction

L'ingénierie des systèmes est une approche multidisciplinaire définie scientifiquement. Elle s'appuie sur des méthodes, des outils, des normes et standards, et des connaissances facilitant la maîtrise du cycle de vie des systèmes complexes. Cela inclue l'idéation, la spécification, la conception, la validation, la réalisation, jusqu'au démantèlement.

Selon l'INCOSE (www.incose.org), l'ingénierie système se caractérise par un ensemble de préoccupations présentées dans la Figure ci-dessous. Ces préoccupations incluent notamment les aspects informatiques, logiciels et matériels, mais pas seulement aussi les lois physiques de façon générale.



L'ingénierie système
selon l'INCOSE (Credit to INCOSE).

Quel que soit le type de système, l'ingénierie informatique, c'est-à-dire logicielle et matérielle, est étroitement liée à l'ingénierie système. De l'ingénierie des exigences à l'IVVQ (Intégration, Vérification, Validation, et Qualification), les pratiques de modélisation (fonctionnelle, non-fonctionnelle, dysfonctionnelle) formelles ou non formelles, sont de plus en plus utilisées.

Les systèmes peuvent être ou non embarqués. Dans le cas de systèmes embarqués, il faut en particulier résoudre des problèmes d'optimisation, d'encombrement, de poids, d'énergie, etc. Entrent dans cette catégorie, les systèmes dits cyber-physiques (CPS, [1]) car ils contrôlent un environnement physique comme les systèmes mécatroniques, les systèmes dits IoT ([2]), les système de type « cloud computing » [3] ou encore « edge computing » [4].

Notons que cette feuille de route se focalise sur la maîtrise du développement des systèmes complexes. Les systèmes sont complexes de par leur hétérogénéité, leur nombre important de constituants en interactions multiples et dont la compréhension ou le développement demande de mettre en œuvre un grand nombre de connaissances dépassant les capacités d'un seul individu, nécessitant ainsi un collectif.

Le développement de ces systèmes fait l'objet d'une compétition économique intense générant des contraintes toujours plus fortes sur les moyens de développement et de production. Parmi ces contraintes, reviennent le plus souvent l'innovation, la réduction des coûts et des temps de développement et l'amélioration de la qualité des produits. On voit également une demande toujours plus accrue de fonctionnalités d'interaction et donc de communication entre les différents systèmes qui font le quotidien des utilisateurs ou celui des entreprises.

On voit également apparaître un glissement de « l'utilisateur » au « client » qui a également pour conséquence d'étendre l'ingénierie au-delà des parties prenantes habituelles par la prise en compte des fonctions marketing et commerciales.

Par ailleurs, le fonctionnement des systèmes complexes doit se conformer de plus en plus à un nombre croissant de contraintes normatives et/ou de régulation. On peut citer entre autre les normes de sûreté (par exemple, ISO 26262, DO-178C, CEI 61513 et CEI 62061) et de sécurité, ou encore les normes anti-pollution (par exemple, les normes d'émissions « Euro » pour les moteurs thermiques).

Néanmoins, quand on s'intéresse à la complexité des systèmes, il est important de faire la distinction entre complexité essentielle et complexité accidentelle telles que définies par Brooks dans [5] :

La complexité essentielle est relative à la nature même du système. Elle est inévitable et pour les raisons détaillées ci-avant (compétition et normes), elle est en expansion dans le domaine des systèmes cyber-physiques.
La complexité accidentelle est celle qui naît soit de l'utilisation de mauvais moyens de développement (par exemple, des outils inadaptés ou des méthodes de travail trop généralistes pour être efficaces), soit d'une mauvaise utilisation des moyens disponibles (par exemple, utiliser un outil sans lire son mode d'emploi peut s'avérer contre-productif). La complexité accidentelle liée au développement des systèmes est donc évitable, si l'on utilise les bons moyens de la bonne méthode.

La raison principale qui explique le phénomène de croissance de la complexité essentielle des systèmes est ce besoin vital d'innovation. Cela se traduit principalement par plus de fonctions, plus « d'intelligence » dans les fonctions, plus d'autonomie, et plus d'interactions (communications). Or, si on augmente le nombre des fonctions et le nombre des interactions entre les fonctions, il est aisé de comprendre qu'on augmente inexorablement la complexité (essentielle) des systèmes. De plus, les interactions entre les différentes fonctions peut-être à l'origine de comportements dît émergents rajoutant un élément supplémentaire de complexité. Le problème actuel est que le niveau de complexité dépasse déjà les capacités humaines à les gérer [Robert N. Charette 2005].

La tendance à considérer des systèmes de systèmes, c'est-à-dire des systèmes composés de sous-systèmes interconnectés de nature différentes, dont la capacité globale est plus importante que la somme de chaque sous-système, ne va pas arranger la situation.

Aller plus loin, nécessite donc de nouveaux moyens, plus puissants et plus efficaces, pour aider les parties prenantes d'un développement afin de gérer et maîtriser cette complexité.

En ce qui concerne la complexité accidentelle, celle-ci est par définition un phénomène que l'on peut éviter, il suffit pour cela de la prévenir. Si elle est la conséquence d'une utilisation inappropriée de langages, méthodes, processus, technologies, et/ou outils, éliminer ce type de complexité revient à mettre en place des artefacts méthodologiques appropriés au domaine d'application et/ou aux différentes préoccupations des acteurs du développement.

Quelle soit accidentelle ou essentielle, la complexité des systèmes et des logiciels est donc un enjeu majeur pour notre société. Dans ce contexte, nous avons organisé la feuille de route sur l'ingénierie système et logiciel autour des thèmes suivants :

- la maîtrise de la complexité et de ses nouvelles formes ;
- l'ingénierie système et développement durable ;
- le rapprochement des moyens et de l'objet des développements ;
- au sujet des évolutions technologiques des outils.

La maîtrise de la complexité & de ses nouvelles formes

Contexte et enjeux

La complexité des systèmes rend leurs développements de plus en plus complexes. L'ensemble des contraintes qui cadre les développements ne fait qu'accroître la complexité. Si jusqu'ici l'expérience d'un architecte système suffisait à assurer la maîtrise de cette complexité, c'est de moins en moins le cas.

Des systèmes de plus en plus ouverts (versus des systèmes fermés dont les interactions avec leur environnement sont connues a priori) sont également à l'origine de nouveaux problèmes de complexité, comme par exemple des comportements émergents, c'est-à-dire des comportements non prévus issus des interactions et encore difficile à anticiper. On peut citer aussi « l'effet rebond » qui est en soit une classe particulière de comportement émergent qu'on aimerait savoir prédire.

L'innovation est devenue le maître-mot pour faire face à la concurrence, et l'intelligence artificielle (IA) apparaît dans ce contexte comme une source miraculeuse d'innovation rendant les systèmes intelligents et autonomes. Cela vient ajouter une dose supplémentaire de complexité. L'introduction de technologie de type IA permet en effet de développer de nouveaux services innovants. Mais cela vient aussi avec son lot de problèmes, parmi lesquels on peut citer, en particulier, la confiance et la frugalité.

D'une part, ce besoin en confiance pose alors la question de proposer de nouvelles méthodes de conception, d'analyse, de caractérisation et de qualification des fonctions, des logiciels et des systèmes à base d'IA. D'autre part, les modules d'IA embarqués dans les systèmes pour les rendre plus intelligents sont généralement très consommateur en données, calcul et énergie. Il est donc nécessaire de développer de nouvelles approches facilitant la maîtrise des ressources à tous les niveaux pour aboutir à des IA éco-responsables : algorithmique, codage et architecture, tout particulièrement pour l'IA embarquée. De plus, l'introduction d'IA dans les systèmes complexes rend leur vérification et leur qualification énormément plus difficile.

Au bout du compte, il n'est pas rare de voir des projets dépasser leur budget, leur planning, ou encore de voir des systèmes en préparation devant être corrigés - « patchés » - pour gérer des défauts de développement.

Les méthodes formelles introduites historiquement pour les systèmes dits critiques comme les centrales nucléaires ou l'avionique, on fait leur preuve sans aucun conteste dans ces systèmes critiques. A titre d'exemple, on peut citer, l'analyse statique de logiciel, la preuve de programme, les techniques de vérification, ou encore de génération automatique de tests, les méthodes d'analyse pour diagnostic. Néanmoins, l'usage des méthodes et technologies formelles reste plutôt confidentiel, souvent restreint à une poignée de spécialistes. Il est pourtant évident que cela devrait changer car de plus en plus de domaines partagent des préoccupations communes avec ces systèmes spécifiques dits critiques et devraient donc progressivement bénéficier de l'intérêt des techniques et des méthodes formelles.

L'intérêt des approches dites d'ingénierie dirigée par les modèles pour faire face à la complexité ne fait également plus débat même si leur mise en œuvre au niveau industriel reste trop limitée. Parmi les freins majeurs, on peut citer les habitudes de travail et la complexité des outils, qu'elle soit réelle ou perçue.

Verrous

Face à cet accroissement grandissant de complexité, l'ingénierie système et logiciel doit évoluer pour lever des verrous parmi lesquels on peut citer :

- la gestion des comportements probabilistes et des incertitudes (par exemple, maîtrise des conditions aux limites, et des hypothèses de modélisation) ;
- l'exploration de l'espace de conception ;
- la simulation hybride discret-continu et multi-échelle ;
- les modèles et les outils spécifiques pour prendre en compte conjointement plusieurs préoccupations, comme la sécurité, la sûreté de fonctionnement, la performance et la qualité ;
- la consolidation et la mise en pratique des principes théoriques de développement par contrat et de compositionnalité ;
- les garanties de propriétés pour des systèmes dynamiques et reconfigurables ;
- l'intégration d'artefacts existants dans de nouveaux systèmes contraints par de nouvelles exigences, par exemple des exigences de cyber-sécurité ;
- la facilitation/accroissement et le déploiement des méthodes et des techniques formelles pour satisfaire les besoins des systèmes à base d'IA pour en améliorer la confiance ;
- l'évolution des règles et des pratiques de certification ;
- l'innovation disruptive en termes de concepts, d'usages et de contraintes de conception architecturale système ;
- le déploiement des méthodes agiles et la gestion de la dette technique afférente – « DevOps » versus ingénierie système et logiciel et maîtrise de la maintenabilité versus gestion des dettes techniques ;
- l'amélioration de la réutilisation et de l'ingénierie de ligne de produits ;
- l'aide à faire les « bons » choix technologiques (par exemple, quelle IA et pour quoi faire ?) dans un domaine aussi effervescent que le numérique ;
- l'expérience utilisateur quant aux outils de modélisation et aux outils d'analyse formelle – il s'agit d'une part d'améliorer les outils eux même, mais aussi de modifier la perception des utilisateurs ou l'acceptabilité des outils ;
- la gestion du nombre croissant des systèmes communicants, soit entre eux, soit avec des infrastructures comme les calculateurs et le stockage dans le nuage ;

- la cohérence des traitements et surtout des données produites et consommées par ces systèmes ;
- l'introduction d'intelligence artificielle dans les systèmes ;
- la formalisation des liens entre ingénierie des systèmes et ingénierie des composants dits d'« intelligence artificielle » ;
- l'introduction de l'intelligence artificielle dans les outils d'ingénierie système.

Note : Bien sûr, cette liste n'est pas exhaustive. Par ailleurs, globalement, le problème de passage à l'échelle se pose pour des outils en apparence simples (gestion de configuration, tout au long du cycle de vie, et en particulier sur les systèmes modifiés après déploiement) ou complexes (outils basés sur des méthodes formelles).

Vers une disparition des frontières entre développement & opération

Contexte et enjeux

Aujourd'hui, l'extension des pratiques, des méthodes et des outils issus de l'embarqué et des systèmes critiques au monde de l'IT (« Technologie de l'Information »), de même que la vague IoT a pour effet de confronter les pratiques d'ingénierie et les contraintes métier du monde de l'IT aux pratiques d'ingénierie et aux contraintes du monde de l'embarqué et des systèmes critiques.

On retiendra comme « tension » résultante, la contrainte d'agilité et la forte réactivité imposée par l'accélération des transformations associées à l'évolution des métiers, confrontée au besoin de maîtrise de la performance sous toutes les formes imposées par les systèmes embarqués.

Ainsi, les approches dites « DevOps » qui deviennent prépondérantes dans le monde du développement logiciel se généralisent-elles au monde de l'ingénierie système. En informatique, les approches dîtes « DevOps » visent le rapprochement des pratiques d'ingénierie liées au développement de logiciels (Dev) et les opérations informatiques (Ops) nécessaires à la gestion de leur cycle de vie. Elles visent à raccourcir le cycle de vie du développement des systèmes et à assurer une livraison continue avec une qualité logicielle élevée (définition inspirée de [6]).

L'interaction des systèmes ne cesse de croître conduisant à ce que les systèmes-de-systèmes soient de plus en plus présents ; les nouvelles organisations « tout numérique » et les nouveaux modèles économiques afférents, tout cela implique de plus en plus de parties prenantes qui devront coopérer de manière cohérente, consistante et efficace et milite pour une suppression progressive des silos fonctionnels.

Verrous

On retiendra entre autre comme défis techniques et méthodologiques principaux les verrous suivants :

- la mise en cohérence des processus d'ingénierie qui sont de fait appliqués par des populations d'ingénieurs de cultures différentes. Le problème réside dans la contradiction de la réduction extrême des cycles d'innovation et de développements poussée par les approches de type « DevOps » confrontée aux évolutions des architectures de systèmes dont le rythme de mise à jour est notamment plus lent ;
- la préservation/maintenance d'architectures de systèmes alors que les méthodes agiles tendent à favoriser l'optimisation locale et la problématique de gestion des dettes technologiques souvent résultantes des approches agiles ;
- la conservation de la maintenabilité en assurant une gestion optimale de la dette technique ;
- la co-évolution et la prise en compte du cycle de vie des composantes d'un système complexe ;
- l'articulation entre spécialisations et interdisciplinarité (dé-silotage technique) ;
- les boucles courtes nécessaires entre toutes les parties prenantes ;
- les données issues du système en « exécution » alimentent l'ingénierie ;
- le DevOps au niveau système ;
- l'introduction de nouvelles technologie de type jumeau numérique,réalité virtuelle et augmentée, métavers, hologramme, etc., pour supporter la co-ingénierie interactive et le travail collaboratif.

Au sujet des évolutions techniques des outils

Contexte et enjeux

L'importance des outils dans le cadre du développement de systèmes et de logiciels complexes reste définitivement sous-estimés. Le développement d'outils pour la maîtrise du développement des systèmes et des logiciels complexes est une industrie en soit qu'il faut soutenir nationalement à l'instar des autres industries vitales pour garantir la compétitivité, l'indépendance et la sécurité de l'industrie française.

Au niveau des outils, qu'il s'agisse d'améliorer leur accessibilité ou d'accroître leur performance, l'un des espoirs majeurs d'amélioration de ces derniers repose sur l'introduction de l'intelligence artificielle dans les outils. On parle alors de « cognification » des outils, au sens de Kevin Kelly dans « The Inevitable: Understanding the 12 Technological Forces That Will Shape Our Future ».

Verrous

Afin de traiter de ces enjeux, on peut citer les verrous suivants :

- souveraineté en termes d'outils de développement ;
- collaboration en travail connecté vs. déconnecté ;
- des outils auto-apprenants, voir autonomes, capable d'interagir avec les ingénieurs et apprendre de leurs pratiques (« bots », « ingénieur virtuel », « agents », etc.) ;
- l'importance de la confiance et de l'explicabilité des IA dans un contexte d'outils autonomes ;
- accessibilité... complexité des outils (no code/low code) ;
- boucle courte entre toutes les parties prenantes ;
- travail conjoint et itératif ;
- ingénierie collaborative ;
- partage des mêmes pratiques et connaissances au travers d'une infrastructure de fédération des données d'ingénierie ;
- gestion des contraintes liées à la propriété intellectuelle (propriété, protection des modèles, des données et des savoir-faire, etc.).

Références :

[1] "Système cyber-physique," *Wikipédia*. Apr. 02, 2021. Accessed: Mar. 21, 2022. [Online]. Available: https://fr.wikipedia.org/w/index.php?title=Syst%C3%A8me_cyber-physique&oldid=181482393

[2] "Internet des objets," *Wikipédia*. Mar. 01, 2022. Accessed: Mar. 21, 2022. [Online]. Available: https://fr.wikipedia.org/w/index.php?title=Internet_des_objets&oldid=191523300

[3] "Cloud computing," *Wikipédia*. Feb. 18, 2022. Accessed: Mar. 21, 2022. [Online]. Available: https://fr.wikipedia.org/w/index.php?title=Cloud_computing&oldid=190951364

[4] "Edge computing," *Wikipédia*. Dec. 01, 2021. Accessed: Mar. 21, 2022. [Online]. Available: https://fr.wikipedia.org/w/index.php?title=Edge_computing&oldid=188457836

[5] Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987, doi: 10.1109/MC.1987.1663532.

[6] "Devops," *Wikipédia*. Feb. 06, 2022. Accessed: Mar. 21, 2022. [Online]. Available: https://en.wikipedia.org/wiki/DevOps

Digital Engineering

Rédacteurs

Sébastien Gérard – CEA
Philippe Mils - Thales
Yves Sorel - Inria
Ramy Iskander - Intento Design

Juin 2022

Systematic
Paris Region Deep Tech Ecosystem