Conclusion

1. Résumé

Le paradigme de la conception orientée objet a permis de faire des progrès énormes en termes de taille et de réussite de projets. Mais, il n'est pas suffisant pour produire des conceptions de qualité professionnelles avec les bonnes propriétés de :

- développabilité,
- testabilité,
- extensibilité,
- maintenabilité,
- réutilisabilité.

Il faut alors se référer au savoir-faire du génie logiciel, accumulé depuis 1968, qui est formalisé, en autres, sous forme de :

1. principes (SOLID),
2. règles de conception,
3. patrons de conception,
4. patrons d'architecture.

Il faut aussi adopter une posture d'artisan du code (Software Craftsmanship), qui va de pair avec l'appropriation de l'approche Agile du développement logiciel, et qui prône l'humilité, l'amour du travail bien fait et le partage. L'artisanat logiciel, c'est aussi un ensemble de pratiques qui participent de la conception de logiciels de qualité, en particulier : le cycle de développement itératif et incrémental, la programmation par pair, le développement dirigé par les tests, le code propre et le DevOps.

2. Ce qui n'a pas été vu

Les deux cours de génie logiciel de première et deuxième années ont pour objectif de présenter les concepts majeurs et les connaissances les plus « théoriques » du génie logiciel. Toutefois, il reste encore beaucoup de champs à découvrir pour balayer le génie logiciel, notamment autour des technologies récentes d'aide au développement. Nous en listons quelques-unes ci-dessous.

2.1. Génération automatique de code et de test

La génération de code source est une opération permettant de générer automatiquement du code source. On pense évidemment ici aux outils comme ChatGPT, Copilot, ou Bard. On peut y ajouter les modèles basés sur des patrons de code (méta-programmation).

Toutefois, ces outils aussi bluffants qu'ils soient, ne sont pas magiques. La génération de code nécessite de la part du développeur une bonne définition du processus système/logiciel et une maîtrise solide de la conception logicielle pour contrôler voire réorienter la production.

2.2. L'intégration continue (CI) et le déploiement continue (CD)

En génie logiciel, CI/CD est la combinaison des pratiques d'intégration continue et de déploiement continu voire de livraison continue.

Le CI/CD comble le fossé entre les activités des équipes de développement et d'exploitation en imposant l'automatisation de la création des tests et du déploiement des applications. Les pratiques DevOps modernes impliquent le développement continu, le test continu, l'intégration continue, le déploiement continu et la surveillance continue des applications logicielles tout au long de leur cycle de vie. La pratique CI/CD constitue l'épine dorsale des opérations DevOps modernes.

2.3. Conteneurisation (*Containerization*)

La conteneurisation a été popularisée par Docker en 2013. Elle consiste à rassembler le code du logiciel et tous ses composants (bibliothèques, frameworks et autres dépendances) de manière à les isoler dans leur propre conteneur.

Le logiciel dans le conteneur peut ainsi être déplacé et exécuté de façon cohérente dans tous les environnements et sur toutes les infrastructures, indépendamment de leur système d'exploitation. Le conteneur fonctionne comme une sorte de bulle, ou comme un environnement de calcul qui enveloppe l'application et l'isole de son entourage. C'est un environnement d'exécution portable complet.

2.4. Correction automatique des bogues (Program repair)

La correction automatique des bogues est la réparation automatique des bogues logiciels sans l'intervention d'un programmeur humain. Elle est également communément appelée génération automatique de correctifs, réparation automatique de bogues ou réparation automatique de programmes. L'objectif typique de ces techniques est de générer automatiquement des correctifs pour éliminer les bogues dans les logiciels sans provoquer de régression logicielle.

2.5. Chaos engineering

Le Chaos Engineering est la discipline de l'expérimentation sur un système distribué afin de renforcer la confiance dans la capacité du système à résister à des conditions de turbulences en production.

Dans le développement de logiciels, la capacité d'un système logiciel donné à tolérer les pannes tout en garantissant une qualité de service adéquate (souvent généralisée sous le nom de résilience) est généralement spécifiée comme une exigence. Cependant, les équipes de développement ne parviennent souvent pas à répondre à cette exigence en raison de facteurs tels que des délais courts ou un manque de connaissance du domaine. L'ingénierie du chaos est une technique permettant de répondre à l'exigence de résilience.

L'ingénierie du chaos peut être utilisée pour atteindre la résilience contre les pannes d'infrastructure, les pannes de réseau et les pannes d'applications.

Ce champ disciplinaire a été popularisé par Netflix avec une mise en place drastique. Directement en production, des ingénieurs coupent une partie des serveurs (par exemple toute une région géographique), sans en avertir le service d'exploitation, et on voit comment l'infrastructure restante est capable d'absorber la charge.

2.6. Fouille de référentiels de logiciels (Mining Software Repositories)

Au sein du génie logiciel, le domaine des référentiels de logiciels d'exploration de données analyse les riches données disponibles dans les référentiels de logiciels, tels que les référentiels de contrôle de version, les archives de listes de diffusion, les systèmes de suivi des bogues, les systèmes de suivi des problèmes, etc. pour découvrir des informations intéressantes et exploitables sur les systèmes logiciels, les projets et l'ingénierie logicielle.

Par exemple, en faisant l'extraction de l'information des changements de code source et de rapports de bogues, on peut recréer l'ensemble du processus de développement d'un projet ou de la mise en production. De nos jours, de nombreux référentiels logiciels à source libre sont disponibles au public, offrant des possibilités pour l'analyse empirique de la mise en production.