Synthèse Génie Logiciel

Groupe 1

- Thibaud GABUS
- Cléry GARDES
- Paul DEMESSINE
- Hugo GRIGORD

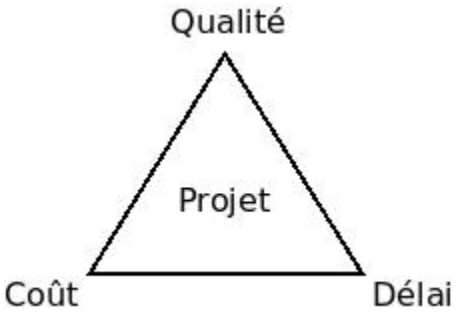
Qu'est-ce que le génie logiciel ?

Le génie logiciel (software engineering) est l'application de principes d'ingénierie au domaine de la création de logiciels. Il consiste à identifier et à utiliser des méthodes, des pratiques et des outils permettant de maximiser les chances de réussite d'un projet logiciel. Le génie logiciel est la réponse aux défis de complexification des logiciels et de l'activité qui vise à les produire.

Les enjeux

Le génie logiciel vise à rationaliser et à optimiser le processus de production d'un logiciel :

- Respect du cahier des charges ;
- Respect des délais de réalisation ;
- Maximisation des performances ;
- Projets évolutifs.



Le génie logiciel couvre une grande dimension :

- Définition de l'architecture du logiciel ;
- Choix de conception ;
- Règles et méthodes de production du code source
- Gestion des versions ;

- Documentation ;
- Organisation de l'équipe et interactions avec le client…

Il faut voir le projet d'un logiciel comme un bâtiment. On n'imagine pas se lancer dans la construction d'un bâtiment sans avoir prévu son apparence, étudié ses fondations et son équilibre, choisi les matériaux utilisés…

Outils de gestion de versions

La gestion de versions est un pilier fondamental du développement logiciel. Elle permet de suivre l'évolution du code source, de collaborer efficacement à plusieurs développeurs et de sécuriser le travail grâce à l'historique de modifications.

Par exemple, Git est aujourd'hui le standard de facto. Il est distribué, rapide et flexible. Il permet à chaque développeur de travailler localement avec un historique complet. Les notions de branches, de commits et de fusions facilitent le travail en parallèle à l'expérimentation.

Des plateformes comme GitHub, GitLab ou Bitbucket enrichissent Git avec des fonctionnalités essentielles :

- Gestion des dépôts distants,
- Revues de code (pull request / merge request),
- Gestion des tickets,
- Intégration continue.

Dans une approche DevOps, la gestion de versions ne se limite plus au code applicatif. On versionne aussi :

- Les scripts d'infrastructure,
- Les pipelines de déploiements,
- La configuration des environnements.

L'objectif est de réduire la frontière entre développement et exploitation, d'automatiser les déploiements et d'améliorer la fiabilité des mises en production.

Gestion de projet

La réussite d'un projet logiciel ne dépend pas uniquement de la technique, mais aussi de l'organisation et du pilotage.

Pour cela, on peut utiliser différentes méthodologies comme le cycle en V ou en Cascade, mais ce sont des approches séquentielles, adaptées à des projets peu flexibles face au changement.

Les méthodes agiles comme Scrum ou Kanban sont les plus utilisées aujourd'hui. Elles permettent l'adaption continue, et prennent en compte les retours utilisateurs et la collaboration. Scrum repose sur des itérations courtes et régulières, les sprints, des rôles définis (Product Owner, Scrum Master, équipe de développement). Ce sont les méthodes à privilégier en gestion de projet.

- Jira est utilisé en entreprise. Il est particulièrement adapté à Scrum et au projets complexes.
- Trello permet de se baser sur des tableaux Kanban, idéal pour des équipes de petite taille ou des projets légers.
- Azure DevOps permet une solution intégrée combinant gestion de backlog, versioning, CI/CD et suivi de projet.

Ces outils permettent de suivre l'avancement, prioriser les tâches, visualiser les dépendances et améliorer la transparence au sein des équipes.

Communication

La communication dans un projet est un facteur clé, en particulier dans des équipes distribuées à travers le monde ou en télétravail.

- Microsoft Teams permet de combiner messagerie, visioconférence, partage de fichiers et intégration avec d'autres outils (Azure DevOps, Office 365)
- Discord est initialement orienté communautés et gaming, il est de plus en plus utilisé pour des équipes techniques grâce à sa fluidité et ses canaux thématiques.
- Slack, très populaire dans les équipes tech, riche en intégrations et automatisations.

Une bonne utilisation de ces outils permet de réduire les silos et de favoriser la collaboration informelle.

Environnements de développement

Aujourd'hui, les IDE modernes offrent bien plus qu'un simple éditeur de code :

- Visual Studio est très complet, avec IntelliSense et un débogueur avancé. Il a des tests intégrés et des outils de profiling. Il est gratuit
- Ride IntelliJ, permet d'avoir des intégrations Git ou autre et des plugins permettant de personnaliser complètement son environnement. Simple d'utilisation mais complexe, il est soumis à license.

Les frameworks structurent le développement et accélèrent la production :

- .NET / ASP.NET pour les applications web, API et desktop,
- Frameworks front-end (React, Angular, Vue),
- Frameworks mobiles et multiplateformes.*

Il faut savoir que le débogage est essentiel pour comprendre le comportement du logiciel. Les outils modernes permettent de fixer des points d'arrêt, d'inspecter la mémoire et analyser les performances. Les tests unitaires contribuent fortement à la qualité et à la maintenabilité du code.

Qualité logicielle

Le choix d'architecture est important pour l'évolutivité et la robustesse des systèmes. Ils permettent de prendre en compte les retours utilisateurs et d'améliorer à travers les versions le logiciel en production.

Parrallèlement, la sécurité est devenue un point essentiel pour gérer les identités et les accès, se protéger contre les vulnérabilités, et sécuriser les pipelines. Elle doit être intégré dès la conception.

Conclusion

Le génie logiciel moderne repose sur un équilibre entre outils techniques, méthodes de travail et collaboration humaine. La gestion de versions, les approches DevOps, les outils de gestion de projet, de communication et les environnements de développement forment un écosystème cohérent au service de la qualité et de la productivité. Au-delà des outils, c'est surtout la capacité des équipes à adopter de bonnes pratiques, à communiquer efficacement et à s'adapter en continu qui fait le succès des projets logiciels.