AGL, GLAO (CASE)

Computer Aided Software Engineering

N° de la lecture individuelle :	4
Semestre	4
Étudiant	DAVID Guillaume, 803_1F
Sujet	AGL (CASE) - Computer Aided Software Engineering

Support théorique

La recherche se base fondamentalement sur les documents présentés ci-dessous ainsi que de la recherche. Des apports, de l'aide à la construction des exemples, et des compréhensions ont également réalisés avec ChatGPT.

Documents

https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel
https://pcsoft.fr/, https://www.embarcadero.com/fr/
https://docs.oracle.com/cd/E24096_01/eacs.pdf
https://www.tutorialspoint.com/software_engineering/case_tools_overview.htm

https://www.sciencedirect.com/topics/computer-science/computer-aided-software-engineering-tool

Table des matières

Introduction

À l'ère de la technologie numérique, le développement de logiciels est devenu un domaine d'activité essentiel, mais aussi complexe. Les exigences des utilisateurs sont de plus en plus pointues, les cycles de développement doivent être rapides et les logiciels eux-mêmes doivent être robustes et évolutifs. Face à ces défis, les équipes de développement ont besoin d'outils et de méthodologies qui leur permettent de naviguer efficacement à travers les phases du développement logiciel, de la conception initiale à la maintenance continue.

Importance des AGL (CASE)

C'est là que le Computer-Aided Software Engineering (CASE) entre en jeu. Le CASE représente une avancée significative dans le domaine du génie logiciel, en offrant un ensemble d'outils et de techniques conçus pour soutenir les développeurs à chaque étape du processus de développement. Ces outils peuvent inclure des environnements de modélisation, des générateurs de code, des outils de test automatisés, des systèmes de gestion de configuration, et bien plus encore.

L'objectif principal du CASE est de simplifier et d'automatiser les tâches répétitives et laborieuses du développement logiciel, tout en offrant un environnement intégré qui favorise la collaboration entre les membres de l'équipe. Grâce à l'utilisation de ces outils, les développeurs peuvent accélérer la production de logiciels tout en maintenant des normes de qualité élevées.

De plus, le CASE permet également une meilleure gestion de projet en fournissant des fonctionnalités de suivi, de planification et de gestion des ressources. Cela aide les chefs de projet à garder une vision claire de l'avancement du projet et à prendre des décisions éclairées pour répondre aux défis potentiels.

Qu’est-ce que le génie logiciel ?

Le génie logiciel, également connu sous le nom d'ingénierie logicielle, est une discipline qui concerne la conception, le développement, la maintenance et la gestion de logiciels de manière **systématique et efficace**. Il s'agit d'appliquer des principes d'ingénierie à la création de logiciels, tout en tenant compte des contraintes liées aux ressources, aux délais et aux exigences des utilisateurs.

- **Processus de développement**
  Le génie logiciel implique l'utilisation de processus systématiques et itératifs pour développer des logiciels. Ces processus comprennent des activités telles que la collecte des exigences, la conception, le codage, les tests et la maintenance.

- **Gestion de projet**
  Il s'agit de gérer efficacement les ressources, les délais et les coûts associés au développement logiciel. Cela comprend la planification des tâches, l'allocation des ressources, la surveillance de l'avancement du projet et la gestion des risques.

- **Qualité du logiciel**

Le génie logiciel vise à produire des logiciels de haute qualité qui répondent aux besoins des utilisateurs de manière fiable, sécurisée et efficace. Cela implique la mise en œuvre de techniques de développement et de tests appropriées pour garantir la qualité du produit final.

- **Modularité et réutilisabilité**
  Une approche fondamentale du génie logiciel est de concevoir des logiciels modulaires et réutilisables. Cela permet de simplifier le développement, de réduire les erreurs et de faciliter la maintenance à long terme.

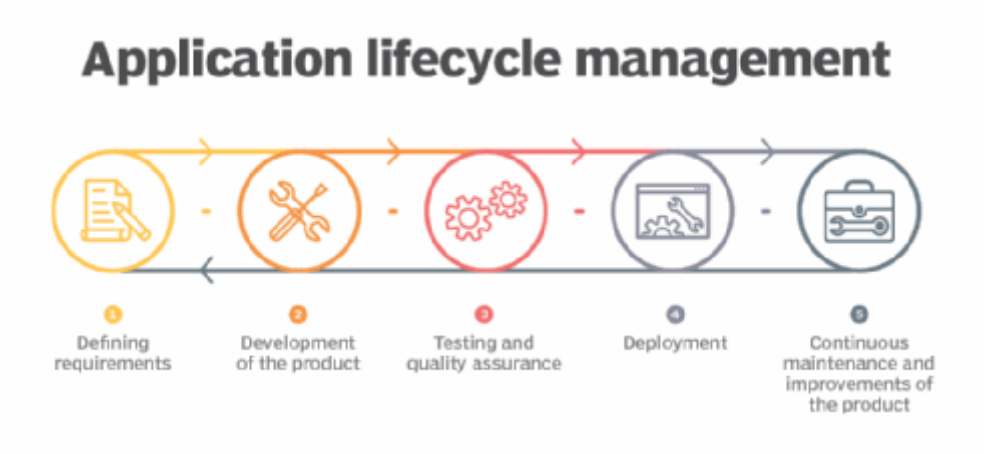
- **Évolution continue**
  Le génie logiciel reconnaît que les logiciels sont sujets à des changements et des évolutions constants. Par conséquent, il est important de concevoir des systèmes qui peuvent facilement s'adapter aux nouvelles exigences et aux évolutions technologiques.

Cycle de vie du développement logiciel et culture DevOps

Application Lifecycle Management

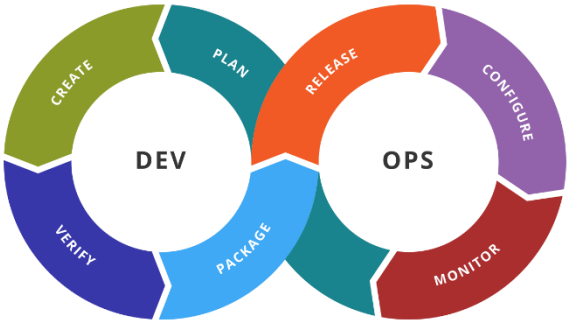
L'Application Lifecycle Management (ALM) est un processus qui couvre tout le cycle de vie d'une application, depuis sa conception initiale jusqu'à son arrêt. Il comprend des activités telles que la gestion des exigences, la modélisation, le développement, les tests, le déploiement, la maintenance et la gestion des versions.

ALM vise à fournir une vision globale et cohérente du processus de développement, en permettant aux équipes de travailler de manière synchronisée sur toutes les phases du cycle de vie de l'application.



DevOps

DevOps est une culture, une philosophie et un ensemble de pratiques visant à améliorer la collaboration entre les équipes de développement (Dev) et d'opérations informatiques (Ops). L'objectif principal de DevOps est de raccourcir le cycle de développement logiciel, d'accélérer le déploiement des applications et d'améliorer la fiabilité des infrastructures logicielles. Pour y parvenir, DevOps favorise l'automatisation des processus, l'intégration continue, le déploiement continu, et une approche axée sur la collaboration et la responsabilité partagée.



ALM / DevOps

Le lien entre ALM et DevOps réside dans le fait qu'ils partagent des objectifs communs, notamment l'amélioration de la collaboration, l'automatisation des processus et la livraison continue. ALM fournit un cadre global pour gérer l'ensemble du cycle de vie de l'application, tandis que DevOps offre des pratiques et des outils spécifiques pour accélérer et améliorer la livraison des logiciels. Ensemble, ALM et DevOps permettent aux organisations de créer un processus de développement plus efficace, agile et orienté vers la livraison de valeur aux utilisateurs finaux.

Concept fondamentaux des AGL (CASE)

Objectifs

Le Génie Logiciel Assisté par Ordinateur ou Computer Aided Software Engineering (CASE) est un ensemble d'outils et de techniques qui simplifient le processus de développement logiciel, de la planification au déploiement. Son utilisation vise à automatiser les tâches et à aider les développeurs à créer des logiciels de haute qualité plus rapidement et avec moins d'erreurs.

Les objectifs principaux visés par le CASE sont :

- Accroître l'efficacité en réduisant le temps et les efforts nécessaires au développement.
- Améliorer la qualité du logiciel en réduisant les erreurs.
- Faciliter la collaboration entre les membres de l'équipe de développement.
- Automatiser la documentation pour gagner du temps et réduire les erreurs.

Concepts

1. **Accroître l'efficacité en réduisant le temps et les efforts nécessaires au développement**
   - Les outils CASE fournissent des fonctionnalités telles que la modélisation visuelle, les générateurs de code automatiques et les environnements de développement intégrés qui permettent aux développeurs de travailler de manière plus efficace.
   - La modélisation visuelle permet aux développeurs de conceptualiser et de concevoir rapidement des systèmes logiciels en utilisant des diagrammes et des notations visuelles, ce qui accélère le processus de conception.
   - Les générateurs de code automatiques traduisent les modèles et les spécifications en code source fonctionnel, réduisant ainsi la quantité de code manuellement écrit et minimisant les erreurs potentielles.

- Les environnements de développement intégrés offrent des fonctionnalités telles que l'autocomplétion, la détection d'erreurs en temps réel et la gestion des versions, ce qui permet aux développeurs de travailler de manière plus productive et organisée.

2. **Améliorer la qualité du logiciel en réduisant les erreurs**
   - Les outils CASE fournissent des fonctionnalités de validation et de vérification intégrées qui permettent aux développeurs de détecter et de corriger les erreurs dès le début du processus de développement.
   - Les environnements de développement intégrés offrent des fonctionnalités de débogage avancées, telles que le suivi des variables, les points d'arrêt et les outils de profilage, qui aident les développeurs à identifier et à résoudre les problèmes plus rapidement.
   - La modélisation visuelle permet aux développeurs de mieux comprendre les exigences et les interactions du système, ce qui réduit les risques d'erreurs de conception et de spécification.

3. **Faciliter la collaboration entre les membres de l'équipe de développement** :
   - Les outils CASE offrent des fonctionnalités de collaboration en temps réel, telles que le partage de modèles, la gestion des tâches et les commentaires intégrés, qui permettent aux membres de l'équipe de travailler ensemble de manière synchronisée.
   - Les environnements de développement intégrés offrent des fonctionnalités de gestion de code source et de suivi des modifications qui facilitent la collaboration sur le code et la résolution des conflits.
   - Les outils CASE fournissent des fonctionnalités de documentation automatique et de gestion des exigences, ce qui aide à aligner les membres de l'équipe sur les objectifs du projet et à garantir une compréhension commune des spécifications.

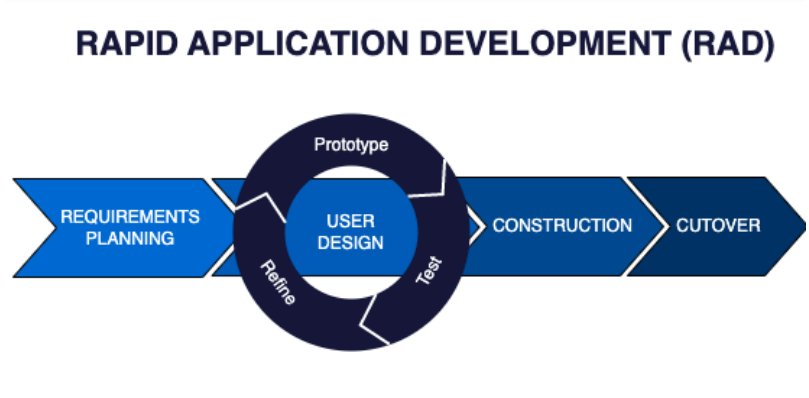
4. **Automatiser la documentation pour gagner du temps et réduire les erreurs** :
   - Les outils CASE génèrent automatiquement une documentation précise et détaillée, y compris des spécifications, des diagrammes de conception et des rapports de test, ce qui réduit la charge de travail des développeurs et minimise les risques d'erreurs humaines.
   - La documentation générée automatiquement est toujours synchronisée avec le code source et les modifications apportées au système, assurant ainsi l'exactitude et la pertinence de la documentation tout au long du cycle de vie du logiciel.
   - Les outils CASE fournissent des fonctionnalités de suivi des exigences et de gestion des versions qui permettent aux développeurs de suivre l'évolution du système et de documenter les changements de manière transparente.

L’utilisation du CASE dans le développement de logiciels permet aux développeurs de produire des logiciels de qualité supérieure en moins de temps et avec moins d'erreurs, ce qui réduit les coûts et améliore la satisfaction des clients.

Techniques de développement logiciel

Le RAD (Rapid Application Development)

Le RAD (Rapid Application Development) est une approche de développement logiciel qui met l'accent sur la rapidité et l'itération dans le processus de création d'applications. Contrairement aux méthodologies traditionnelles de développement logiciel, qui suivent généralement une approche séquentielle et rigide, le RAD favorise des cycles de développement rapides et itératifs.



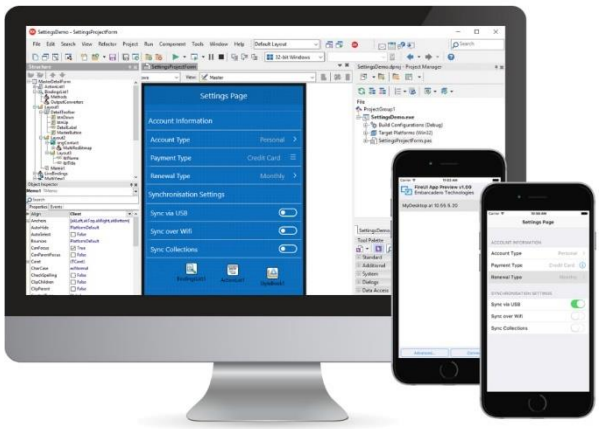
- **Prototypage rapide**
  Le RAD encourage la création rapide de prototypes fonctionnels de l'application. Ces prototypes permettent aux développeurs et aux clients de visualiser et de tester rapidement différentes fonctionnalités et interfaces utilisateur, ce qui favorise un processus de développement itératif.

- **Implication étroite des utilisateurs finaux**
  Les utilisateurs finaux sont impliqués tout au long du processus de développement RAD. Leurs retours d'expérience sont pris en compte dès les premières étapes du développement, ce qui permet d'adapter rapidement l'application à leurs besoins et attentes.

- **Itérations rapides**
  Le RAD favorise des cycles de développement courts, généralement de quelques semaines à quelques mois, au cours desquels des versions fonctionnelles de l'application sont développées, testées et améliorées de manière continue.

- **Utilisation intensive d'outils de développement visuel**
  Le RAD repose sur l'utilisation d'outils de développement visuel qui permettent aux développeurs de créer rapidement des interfaces utilisateur et des fonctionnalités en utilisant des éléments visuels plutôt que du code traditionnel.

- **Flexibilité et adaptabilité**
  Le RAD est particulièrement adapté aux projets où les exigences sont susceptibles de changer fréquemment ou lorsque des délais serrés nécessitent une livraison rapide de l'application.



Outils et fonctionnalités AGL (CASE)

Les Génies Logiciels Assistés par Ordinateur (Computer-Aided Software Engineering, CASE) proposent une variété d'outils conçus pour faciliter et automatiser divers aspects du processus de développement logiciel.

Les CASE propose des fonctionnalités

1. **Outils de modélisation**
   Ces outils permettent de créer des représentations visuelles des différentes parties d'une application logicielle, facilitant ainsi la compréhension et la communication des concepts et des structures.

2. **Outils de génération de code**
   Ils automatisent la création de code source à partir de modèles ou de spécifications, accélérant ainsi le processus de développement et réduisant les erreurs humaines.

3. **Outils de gestion de projet**
   Ces outils aident à planifier, organiser et suivre les projets logiciels, en fournissant des fonctionnalités de suivi des tâches, de gestion des ressources et de génération de rapports sur l'avancement du projet.

4. **Outils de test et de validation**
   Ils permettent d'automatiser les tests logiciels, de vérifier la conformité aux exigences et d'assurer la qualité du logiciel.

5. **Outils de documentation automatique**
   Ces outils génèrent automatiquement une documentation détaillée du logiciel, y compris des spécifications, des diagrammes et des rapports, ce qui facilite la maintenance et la compréhension du système.

6. **Outils de gestion des configurations**
   Ils facilitent le suivi des versions du logiciel, la gestion des modifications et la coordination du travail entre les membres de l'équipe.

7. **Outils de suivi des exigences**
   Ces outils permettent de capturer, suivre et gérer les exigences du logiciel tout au long du cycle de vie du projet.

8. **Outils de collaboration en équipe**
   Ils facilitent la communication et la collaboration entre les membres de l'équipe de développement, en offrant des fonctionnalités telles que le partage de documents, les discussions en ligne et la gestion des tâches.

9. **Outils de gestion des risques**
   Ils aident à identifier, évaluer et gérer les risques potentiels associés au développement et à la maintenance du logiciel.

10. **Outils de diagrammes UML (Unified Modeling Language)**
    Ils fournissent des fonctionnalités pour créer des diagrammes conformes aux normes UML, ce qui permet de modéliser visuellement les aspects structurels et comportementaux du logiciel.

11. **Outils de gestion des tests**
    Ils facilitent la planification, l'exécution et le suivi des activités de test logiciel, garantissant ainsi la qualité et la fiabilité du produit final.

12. **Outils de gestion des versions**
    Ces outils permettent de gérer les différentes versions du code source et des documents associés, en assurant la traçabilité et la cohérence des modifications apportées au logiciel.

13. **Outils de diagrammes d'architecture**
    Ils permettent de créer des représentations visuelles de l'architecture logicielle, facilitant ainsi la compréhension et la communication des concepts et des décisions architecturales.

14. **Outils de cartographie des processus**
    Ils permettent de modéliser et de visualiser les processus métier, facilitant ainsi la compréhension et l'optimisation des flux de travail.

15. **Outils de gestion de workflow**
    Ils automatisent les processus et les flux de travail, en fournissant des fonctionnalités de suivi, de routage et de gestion des tâches.

16. **Outils de modélisation de bases de données** : Ils permettent de concevoir et de gérer des bases de données, en fournissant des fonctionnalités de modélisation visuelle et de génération de scripts SQL.

17. **Outils de simulation**
    Ils permettent de simuler le comportement du logiciel dans des conditions spécifiques, aidant ainsi à évaluer la performance et la fiabilité du système.

18. **Outils de prototypage interactif**
    Ils facilitent la création rapide de prototypes fonctionnels de l'interface utilisateur, permettant ainsi de recueillir rapidement les retours des utilisateurs et de valider les concepts de conception.

19. **Outils de surveillance des performances**
    Ils fournissent des fonctionnalités pour surveiller et analyser les performances du logiciel en temps réel, permettant ainsi d'identifier et de résoudre les problèmes de performance.

20. **Outils de gestion des déploiements**
    Ils automatisent le processus de déploiement du logiciel, en fournissant des fonctionnalités pour planifier, exécuter et surveiller les déploiements de manière efficace et fiable.

21. **Outils de requêtage**
    Ils permettent d'interroger et de manipuler des bases de données de manière intuitive, en fournissant des fonctionnalités de création et d'exécution de requêtes SQL.

22. **Outils de génération de rapport**
    Ils permettent de créer des rapports personnalisés à partir des données générées par le système, facilitant ainsi la communication et la prise de décision.

Intégration et abstraction

L'intégration de ces outils au sein d'un même environnement facilite le processus de développement de plusieurs manières :

- **Connaissance de l'environnement global**
  Cette intégration offre une vue d'ensemble cohérente du projet, permettant aux développeurs de naviguer efficacement entre les différentes fonctionnalités et de comprendre rapidement les relations entre les composants du système.

- **Éviter les passerelles**
  En regroupant les outils, elle élimine la nécessité de recourir à des interfaces distinctes pour passer d'un outil à l'autre. Cela simplifie le flux de travail en réduisant la complexité associée à la gestion de multiples interfaces utilisateur.

- **Abstraction de niveau supérieur**
  Les outils intégrés fournissent souvent des abstractions plus élevées, ce qui permet aux développeurs de se concentrer sur les aspects conceptuels et fonctionnels du logiciel plutôt que sur les détails d'implémentation. Cette approche facilite la création et la compréhension des modèles logiciels.

- **Databinding**

L'intégration du databinding automatise la liaison des données avec les éléments d'interface utilisateur, simplifiant ainsi la gestion des données et améliorant l'expérience de développement en réduisant la nécessité de synchroniser manuellement les données et l'interface utilisateur.

Le concept de databinding et d'abstraction sont souvent forts dans ce type de logiciel car ils permettent de simplifier la manipulation et la gestion des données ainsi que les objets du projet.

En utilisant le databinding, les développeurs peuvent lier les données à des éléments d'interface utilisateur sans avoir à écrire de code de liaison spécifique, ce qui réduit considérablement la quantité de code nécessaire et facilite la maintenance du logiciel.

De plus, le databinding permet une mise à jour automatique de l'interface utilisateur en réponse aux changements des données, raccourcissant le temps de développement.

Parallèlement, l'abstraction permet de simplifier la complexité du système en masquant les détails d'implémentation sous-jacents et en fournissant des interfaces et des modèles de haut niveau. Cela permet aux développeurs de se concentrer sur les aspects fonctionnels et conceptuels du logiciel, plutôt que sur les détails techniques, ce qui favorise une conception logicielle plus claire et modulaire.

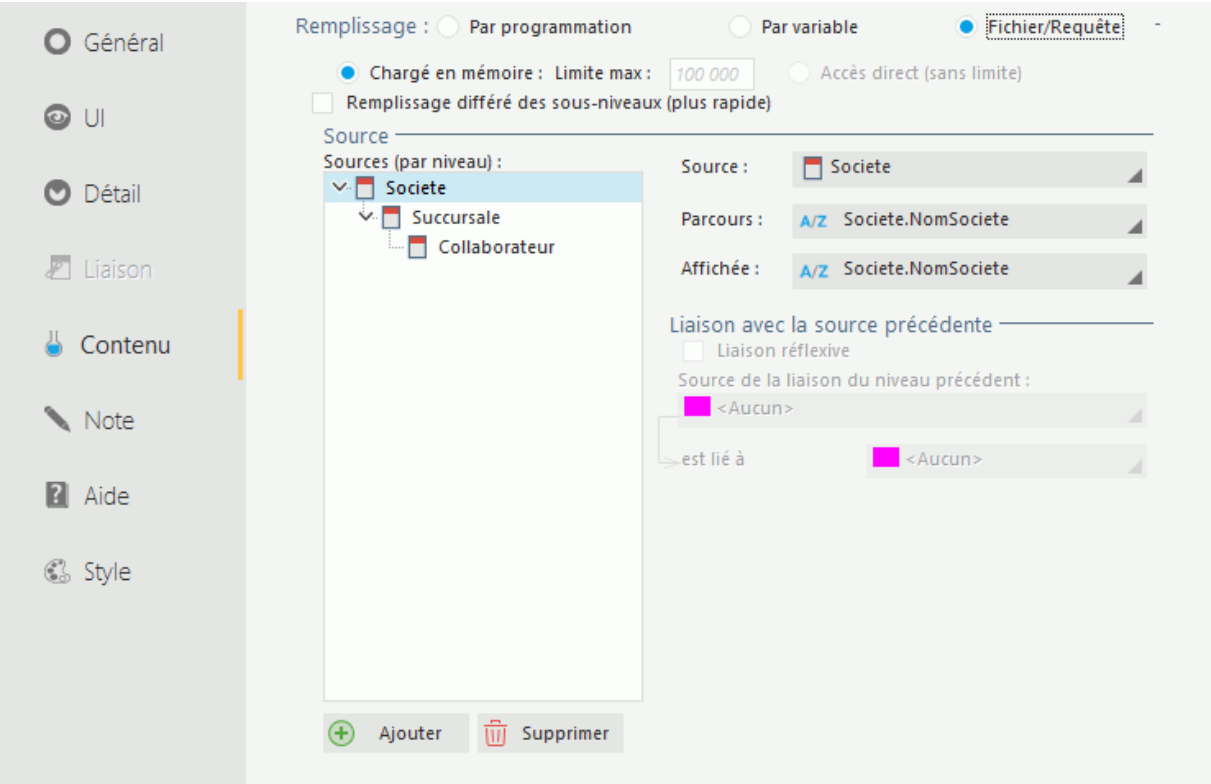
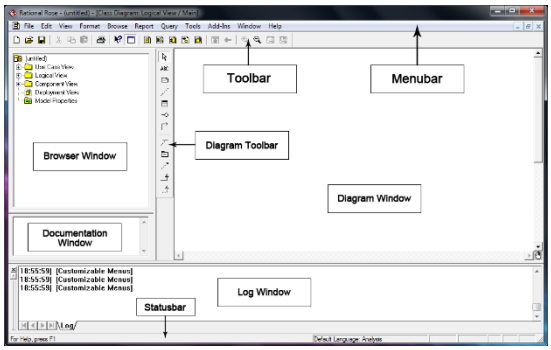


Figure 1- Exemple de databinding (tableau lié à l'analyse)

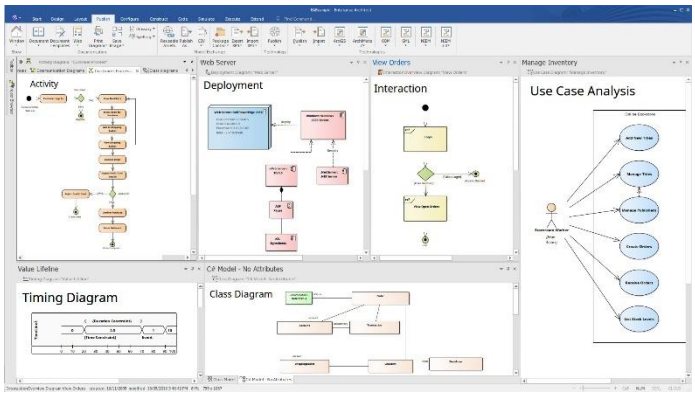
Exemple d'AGL / CASE

Quelques exemples d'AGL (CASE) : xcfax y

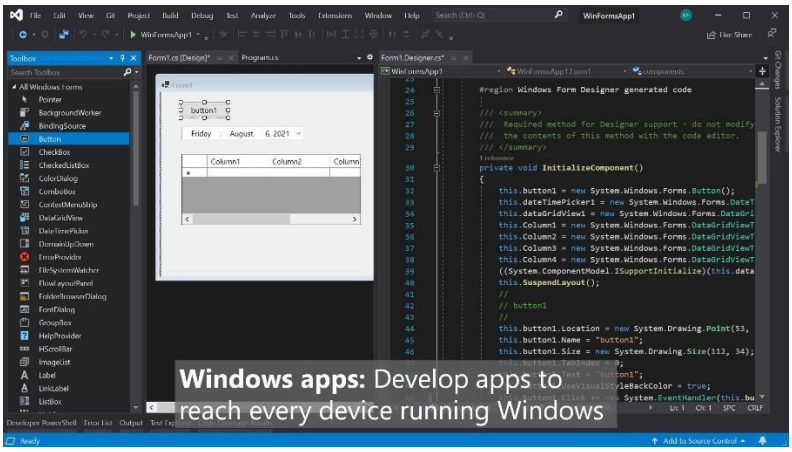
- **Rational Rose d'IBM**
  Rational Rose est un outil de modélisation UML (Unified Modeling Language) largement utilisé pour la conception logicielle. Il offre des fonctionnalités avancées de modélisation pour aider les développeurs à concevoir et à documenter efficacement leurs projets logiciels.



- **Enterprise Architect de Sparx Systems**
  Enterprise Architect est une plateforme de modélisation et de conception logicielle qui prend en charge une large gamme de langages de modélisation et de méthodologies de développement. Il offre des fonctionnalités de modélisation UML, de simulation, de génération de code et de gestion des exigences.



- **Visual Studio**
  Visual Studio est un environnement de développement intégré (IDE) largement utilisé pour le développement d'applications logicielles sur la plateforme Microsoft. Il prend en charge plusieurs langages de programmation tels que C#, VB.NET et C++, et offre des fonctionnalités avancées pour la conception, le débogage et le déploiement d'applications.

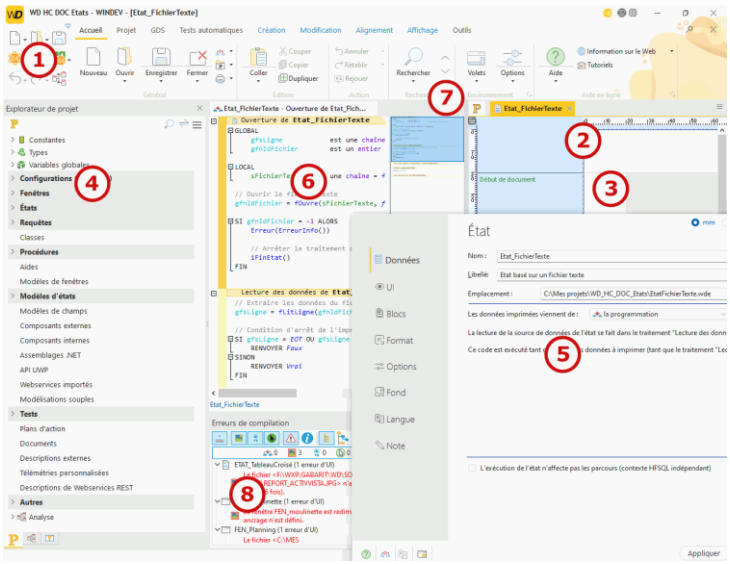


- **NetBeans**
  NetBeans est un IDE open source populaire utilisé pour le développement d'applications Java et d'autres langages de programmation. Il offre des fonctionnalités de développement avancées telles que la prise en charge de la modularité, des outils de refactoring et une intégration étroite avec les frameworks Java.

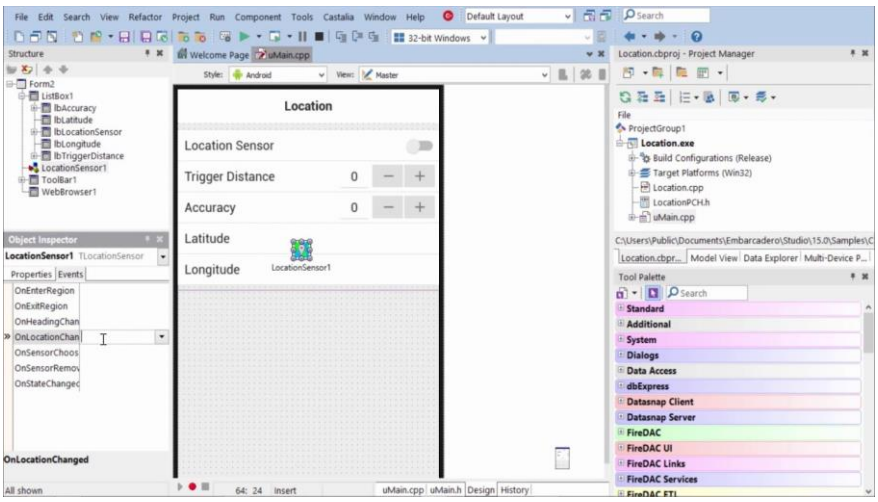
- **Eclipse**
  Eclipse est un autre IDE open source largement utilisé pour le développement d'applications logicielles dans divers langages de programmation, notamment Java, C/C++, PHP et Python. Il est extensible grâce à un large éventail de plugins, ce qui permet aux développeurs de personnaliser leur environnement de développement en fonction de leurs besoins.

- **Visual Paradigm**
  Visual Paradigm est un outil de modélisation UML et de conception logicielle qui offre des fonctionnalités avancées pour la création de diagrammes de classes, de diagrammes de séquence, de diagrammes de flux de données, etc. Il prend en charge la génération de code dans plusieurs langages de programmation et offre des fonctionnalités de collaboration en équipe.

- **Suite PCSoft**
  WinDev est un environnement de développement intégré (IDE) développé par PC SOFT, principalement utilisé pour le développement d'applications Windows. Il offre des fonctionnalités avancées pour la création d'applications de bureau, d'applications web et d'applications mobiles, avec une approche de développement rapide.



- **Embarcadero (anciennement CodeGear et Borland)** : Embarcadero propose plusieurs outils de développement logiciel, notamment Delphi et C++Builder. Delphi est un environnement de développement intégré (IDE) pour le langage de programmation Object Pascal, largement utilisé pour le développement d'applications de bureau, web et mobiles. C++Builder est un IDE pour le développement d'applications en C++ avec une prise en charge pour le développement multiplateforme. Ces outils offrent une gamme complète de fonctionnalités de développement, de conception d'interfaces utilisateur et de déploiement pour les développeurs professionnels.



Avantages / Inconvénient des ALG (CASE)

Les Génies Logiciels Assistés par Ordinateur (CASE) sont des ensembles d'outils et de techniques utilisés pour concevoir, développer et maintenir des logiciels de manière plus efficace. Ils offrent une variété d'avantages, mais ils peuvent également présenter quelques inconvénients.

Avantages

- **Amélioration de la productivité**

Les CASE automatisent de nombreuses tâches du processus de développement, telles que la génération de code, la création de diagrammes, la documentation et les tests. Cela permet aux développeurs de consacrer plus de temps à des activités à plus forte valeur ajoutée, comme la conception et l'optimisation des performances, augmentant ainsi la productivité globale de l'équipe.

- **Meilleure qualité du logiciel**
  En utilisant des outils de modélisation visuelle et des générateurs de code automatiques, les développeurs peuvent réduire les erreurs humaines et les bugs logiciels. De plus, les fonctionnalités de test automatisé intégrées dans certains CASE permettent de détecter et de corriger les erreurs plus tôt dans le processus de développement, ce qui améliore la qualité globale du logiciel.

- **Facilitation de la collaboration**
  Les CASE offrent des fonctionnalités de collaboration en temps réel qui permettent aux membres de l'équipe de travailler ensemble de manière transparente, indépendamment de leur emplacement géographique. Les développeurs peuvent partager des informations, des documents et des ressources facilement, ce qui favorise une meilleure communication et une collaboration efficace.

- **Gestion efficace des changements**
  Les outils de suivi des versions et de gestion des configurations inclus dans les CASE permettent aux développeurs de suivre les modifications apportées au code et de gérer les différentes versions du logiciel de manière efficace. Cela facilite la collaboration au sein de l'équipe et garantit l'intégrité du code source tout au long du cycle de vie du projet.

- **Documentation automatisée**
  Les CASE génèrent automatiquement une documentation détaillée du logiciel, y compris des spécifications, des diagrammes de conception, des rapports de test et des manuels d'utilisation. Cette documentation est constamment mise à jour en fonction des modifications apportées au code, ce qui facilite la maintenance et la compréhension du système pour les développeurs et les parties prenantes.

Inconvénients

Bien évidemment, ils ne sont pas parfait et présentent de nombreux inconvénient.

- **Coût initial élevé**
  La mise en place d'un système CASE peut nécessiter un investissement initial important en termes de coûts de licence, de matériel informatique et de formation du personnel. Pour de nombreuses petites entreprises, cela peut représenter une barrière financière importante à l'adoption des CASE.

- **Complexité de mise en œuvre**
  La configuration et la personnalisation d'un système CASE peuvent être complexes, nécessitant une expertise technique approfondie et une planification minutieuse. Les

équipes de développement doivent investir du temps et des ressources dans la mise en place de l'infrastructure nécessaire et dans la formation des utilisateurs finaux.

- **Dépendance vis-à-vis des fournisseurs**
  Les CASE sont souvent fournis par des entreprises spécifiques, ce qui peut entraîner une dépendance vis-à-vis de ces fournisseurs pour les mises à jour logicielles, le support technique et les services connexes. En cas de problème avec le fournisseur ou de cessation d'activité, cela peut entraîner des interruptions dans le développement logiciel et des difficultés pour migrer vers d'autres plates-formes.

- **Surcharge d'informations**
  La génération automatique de documentation peut parfois entraîner une surcharge d'informations, rendant difficile la navigation et la recherche d'informations pertinentes pour les développeurs. Il est important de mettre en place des processus efficaces pour gérer et organiser la documentation générée par le CASE afin de garantir qu'elle reste utile et pertinente pour l'équipe de développement.

- **Adaptation aux besoins spécifiques**
  Bien que les CASE offrent une gamme de fonctionnalités standard, ils peuvent ne pas répondre pleinement aux besoins spécifiques de chaque projet ou organisation. Cela peut nécessiter des personnalisations importantes ou des compromis dans la manière dont le logiciel est développé, ce qui peut entraîner des frustrations et des inefficacités pour l'équipe de développement.

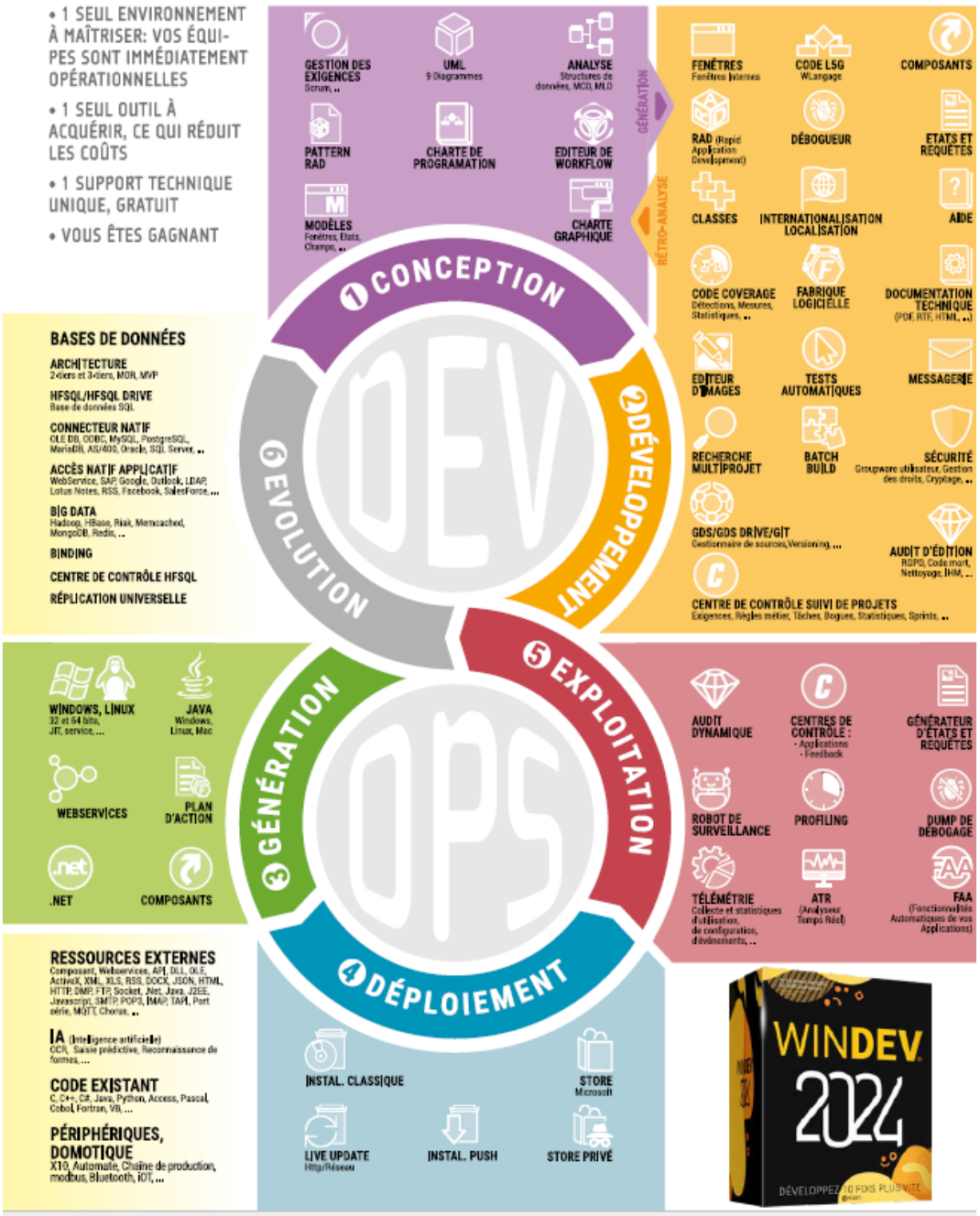
Présentation de l'AGL de PCSoft

Brève présentation de l'AGL PCSoft.

PLATEFORME INTÉGRÉE DE DÉVELOPPEMENT AGILE

DANS WINDEV 2024 TOUT EST INTÉGRÉ

- 1 SEUL ENVIRONNEMENT À MAÎTRISER: VOS ÉQUIPES SONT IMMÉDIATEMENT OPÉRATIONNELLES
- 1 SEUL OUTIL À ACQUÉRIR, CE QUI RÉDUIT LES COÛTS
- 1 SUPPORT TECHNIQUE UNIQUE, GRATUIT
- VOUS ÊTES GAGNANT



Bibliographie

https://fr.wikipedia.org/wiki/G%C3%A9nie_logiciel
https://www.startertutorials.com/uml/wp-content/uploads/2014/06/RationalRose-Interface.png
https://upload.wikimedia.org/wikipedia/commons/9/9d/EA_reflection_simulation.jpg
https://images-eds-ssl.xboxlive.com/image?url=4rt9.lXDC4H_93laV1_eHHFT949fUipzkiFOBH3fAiZZUCdYojwUyX2aTonS1aIwMrx6NUIsHfUHSLzjGJFxxuAehyaZ0tpXbUw9BYCsFptZCsm282wYpBuQPoLxHoFDJFMP2QdngIiQ4Udq_jioZNtG0.Q39wzd2uoqg59kGco-
https://doc.pcsoft.fr/fr-FR/images/image.awp?langid=5&name=EtatEditeur.gif&type=thumb&0
https://i.ytimg.com/vi/4vYzeSXMBr4/maxresdefault.jpg
https://www.pcsoft.fr

**Les images non référencées dans la bibliographie proviennent du livre ou document étudié.*