Évolution et Restructuration du logiciel

Abdelhak-Djamel Seriai

seriai@lirmm.fr

http://www.lirmm.fr/~seriai

Organisation

- **Responsables :**
  - Abdelhak-Djamel Seriai :
    - http://www.lirmm.fr/~seriai/
    - seriai@lirmm.fr
  - Marianne Huchard
    - https://www.lirmm.fr/users/utilisateurs-lirmm/marianne-huchard
    - huchard@lirmm.fr

Organisation

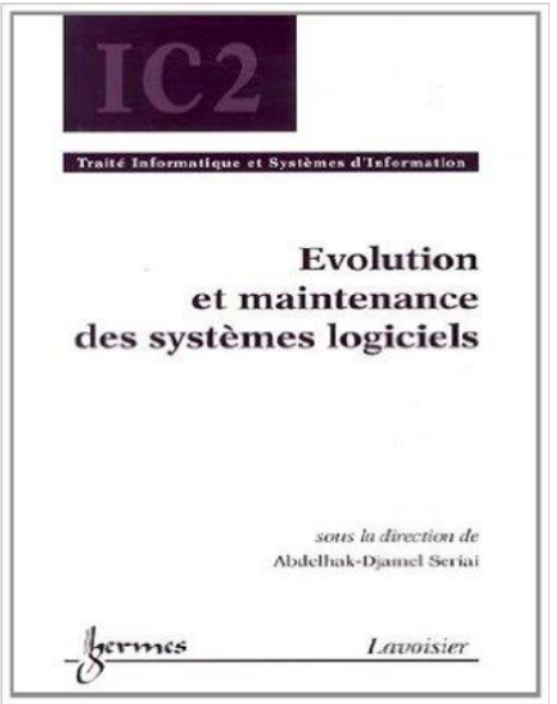
- Cours magistraux + travaux pratiques
- Modalités de contrôle des connaissances
  - TPs
  - QCM

Objectifs de ce cours

- Présenter les aspects essentiels de la maintenance (é volution) des logiciels
  - Aspects techniques de la maintenance
  - Aspects liés à l'organisation et à la gestion des activités de maint

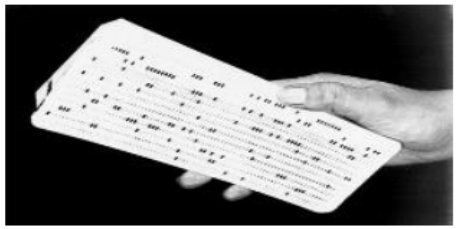
Contenu

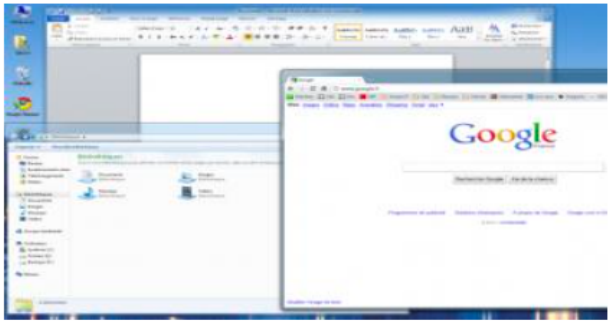
- Concepts et principes généraux de l'évolution logicielle
- Analyse statique et dynamique des logiciels
- Rétro-ingénierie : extraction d'architecture d'un système OO
- Visualisation des logiciels
- Gestion de la variabilité et lignes de produits
- Compréhension des logiciels
- Restructuration et refactoring des logiciels



Motivation

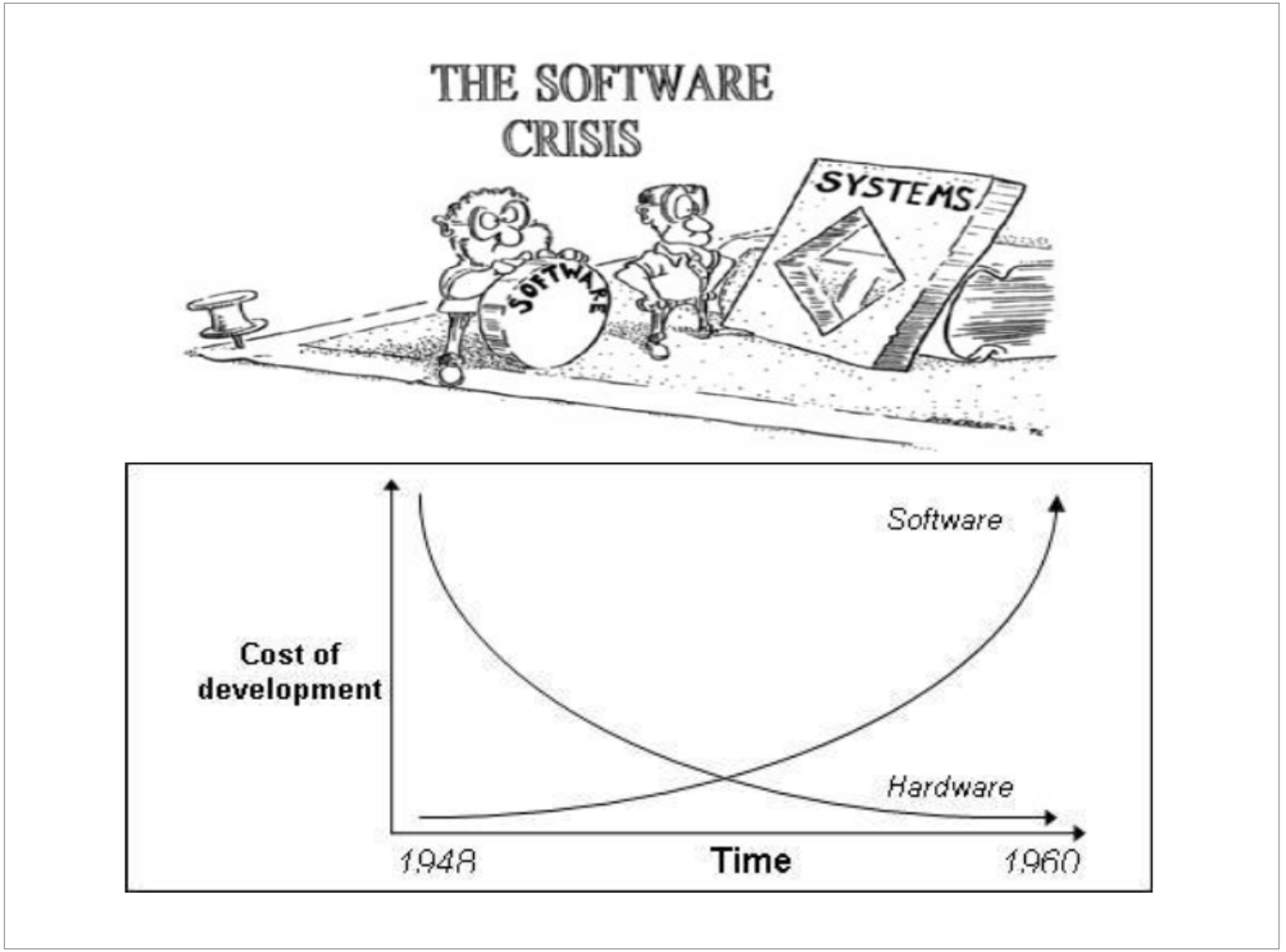
- Logiciel versus matériel : taille, complexité, coût





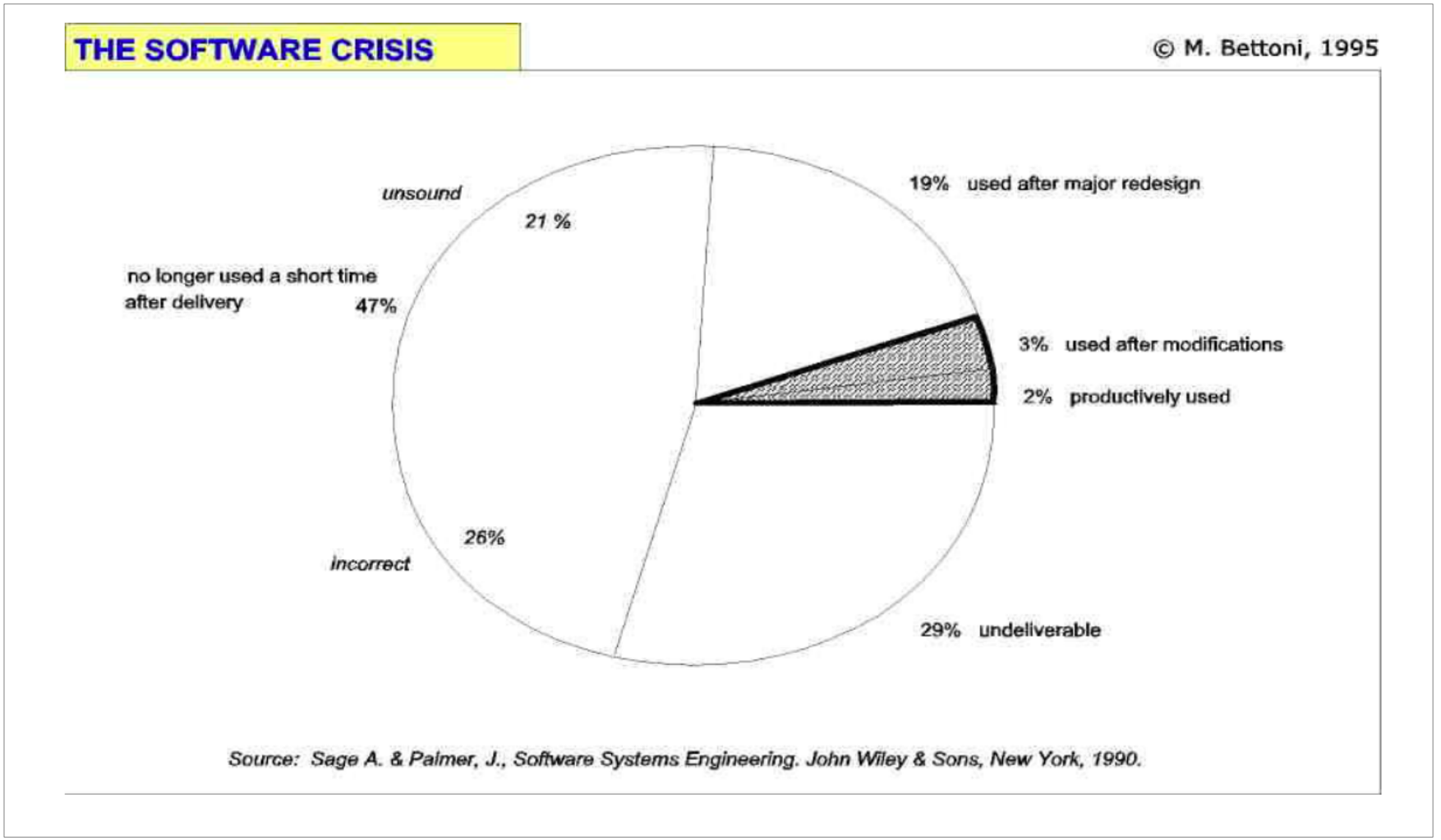
Motivation

- La crise du logiciel en images



Motivation

- La crise du logiciel en images



Motivation

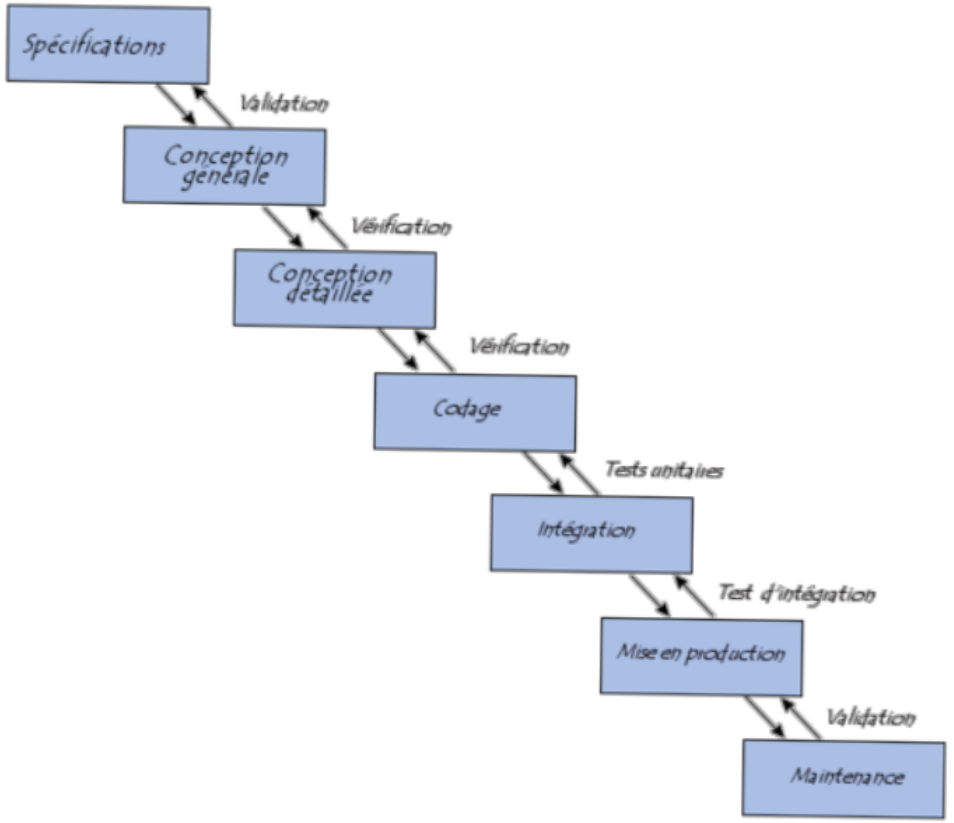
- Quelques exemples de logiciels et leurs tailles

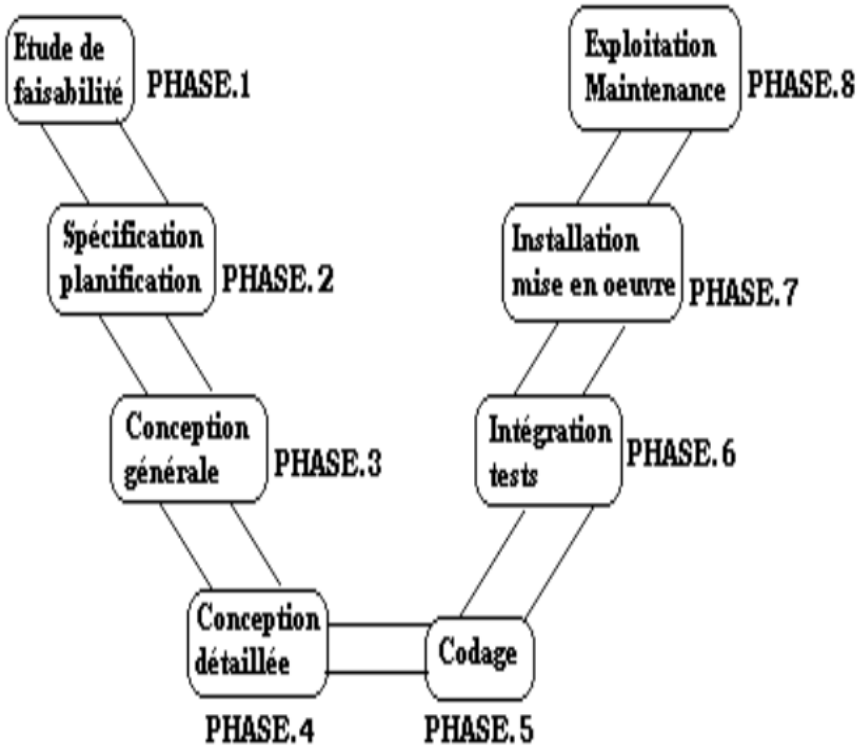
Système	langage	NOP	NOC	kLOC
ArgoUML	Java	141	2442	143
Azureus	Java	457	4734	274
IText	Java	149	1250	80
Jmol	Java	105	1032	85
JDK 1.5	Java	137	4715	160
Moose	Smalltalk	278	994	35
ScrumVM	C++	17	1331	80

- [source Richard Wettel]

Motivation

- Cycle de vie du logiciel, maintenance





Motivation

.Exemples d'activités de maintenance/évolution d'un logiciel

- De nouvelles exigences apparaissent lorsque le logiciel est utilisé ou même quand le logiciel est en développement
- Les changements de l'environnement des affaires
- Nouveaux clients ou les clients existants avec de nouvelles exigences
- Les erreurs doivent être réparées
- Nouveaux ordinateurs et équipements sont ajoutés au système
- La performance ou la fiabilité du système peuvent être améliorées
- De nouvelles technologies sont utilisées (nouvelles normes, de nouveaux OS, nouvelles versions du logiciel, ...)
- Etc.

Définitions : Maintenance/évolution logicielle

- La modification d'un logiciel, après sa livraison, afin de corriger des défaillances, d'améliorer sa performance ou d'autres attributs ou de l'adapter suite à des changements d'environnements --IEEE 610.12, 1993

- Les changements au logiciel et à sa documentation causés par un problème ou le besoin de l'améliorer – ISO 12207, 1995

- La totalité des activités qui sont requises afin de procurer un support, au meilleur coût possible, d'un logiciel. Certaines activités débutent avant la livraison du logiciel, donc pendant la conception initiale, mais la majorité des activités ont lieu après sa livraison finale (quand l'équipe de développement considère qu'elle a terminé son travail) – SWEBOK, 2005

- L'ensemble des activités requises afin de garder le logiciel en état d'opération suite à sa livraison opérationnelle --FIPS, 1984

Les types de maintenance

.Classification de ISO/IEC 14674

Why? / When?	Correction	Enhancement
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive

Les types de maintenance

.Le standard ISO/IEC 14764 distingue quatre catégories:

–Maintenance corrective

.Fixe les différents bogues d'un système.

.Le système peut ne pas fonctionner comme il devrait le faire selon ses plans de conception. L'objectif de la maintenance corrective est donc de localiser les spé cifications originelles afin de déterminer ce que le système était initialement sensé faire.

–Maintenance adaptative

.Fixe les différents changements qui doivent être faits afin de suivre l'environnement du système en perpétuelle évolution.

.Par exemple, le système d'exploitation peut être mis à jour et des modifications peuvent être nécessaires afin de s'accommoder à ce nouveau système d'exploitation.

Les types de maintenance

- Maintenance perfective

  - La maintenance perfective ou d'amélioration comprend tous les changements faits sur un système afin de satisfaire aux besoins de l'utilisateur.

- Maintenance préventive

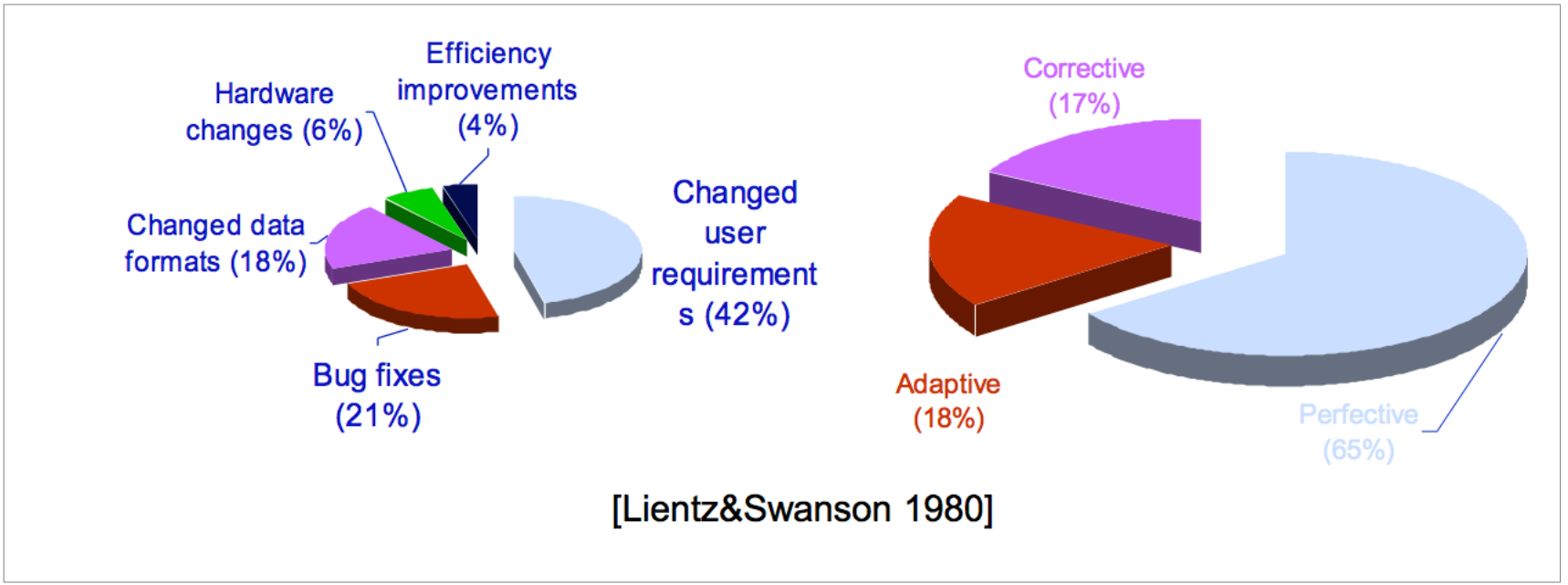
  - La maintenance exécutée afin d'empêcher des problèmes avant qu'ils surviennent.

  - Ce type de maintenance peut être considéré comme très important dans le cas de systèmes critiques.

Les types de maintenance

.Répartition du coût de la maintenance

- Repérer (détecter) des défauts (des erreurs) (17%)
- Adapter un logiciel à différents environnements (machines, OS, etc.) par rapport à son implémentation d'origine (18%)
- Ajouter ou modifier les fonctionnalités du système pour répondre à de nouveaux besoins (65%)



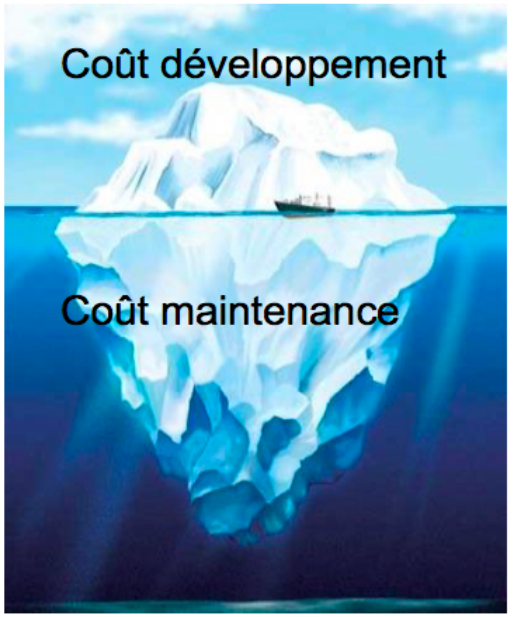
[Lientz&Swanson 1980]

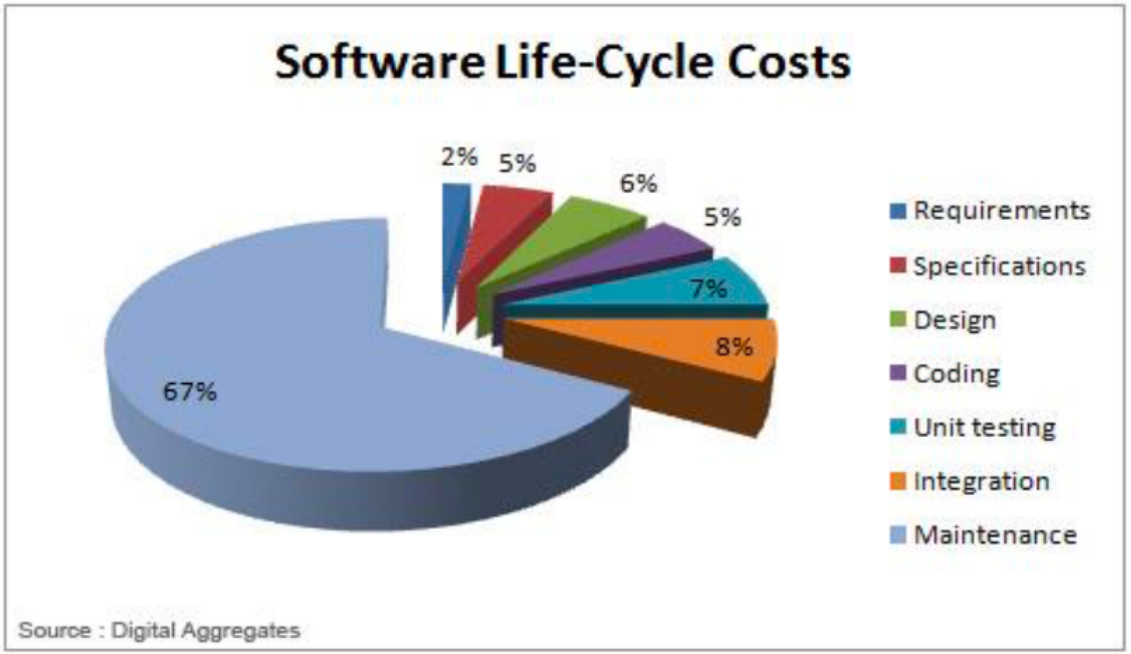
Caractéristiques de la maintenance logicielle

.La maintenance du logiciel est coûteuse

–La majorité du budget de logiciels dans les grandes entreprises est consacrée à l'adaptation des logiciels existants plutôt que de développer un nouveau logiciel

–Les coûts de maintenance sont supérieurs aux coûts estimés avec un facteur compris entre 2 et 100 (selon l'application)





Caractéristiques de la maintenance logicielle

.La maintenance du logiciel est difficile

- Résultat d'une mauvaise conception
- Code mal documenté donc difficulté de compréhension
- Code mal documenté donc difficulté de compréhension
- Activités de maintenance assignées à des programmeurs dé butants
- Non maîtrise de toutes les étapes du processus de maintenance
- Absence d'outils
- Etc.

Caractéristiques de la maintenance logicielle

.Les changements ne peuvent pas être anticipés

–"Le problème fondamental, soutenue par 40 ans d'expérience difficile, est que de nombreux changements réellement requises sont celles que les concepteurs originaux ne peuvent même pas imaginer." [Bennett & Rajlich 2000].

Lois de l'évolution logicielle

- **Les lois de Lehman**
  - Basé sur des études empiriques par rapport à l'évolution sur une période de 30 ans

Lehman M.M. and Belady L.A. (Eds.), 1985 *Software Evolution – Processes of Software Change*, Academic Press, London *(Free download from* wiki.ercim.eu/wg/SoftwareEvolution*)*

M. M. Lehman. *Laws of Software Evolution Revisited*. Lecture Notes in Computer Science 1149, pp. 108-124, Springer Verlag, 1997

Lois de l'évolution logicielle

.La modification continue :

–Un programme utilisé dans un environnement du monde réel doit né cessairement changer sinon il deviendra progressivement de moins en moins utile dans cet environnement.

.La complexité croissante :

–Comme le logiciel est modifié, il devient plus en plus complexe à moins que le travail soit effectué pour réduire la complexité.

.Autorégulation :

–L’évolution des grands programmes est un processus auto- régulateur. Les attributs comme la taille, le temps entre versions et le nombre d’erreurs signalé es sont approximativement invariants pour chaque version du programme.

Lois de l'évolution logicielle

.La stabilité organisationnelle :

–Pendant la vie d'un programme, son taux de développement est approximativement constant et indépendant des ressources qui y sont consacrées.

.La conservation de la familiarité :

–Pendant la vie d'un programme, l'incrément de changement dans chaque version est approximativement constant.

Systèmes hérités (Legacy Systems)

- Les entreprises investissent de larges sommes dans la création de logiciels
  - Les logiciels doivent donc servir plusieurs années par souci de rentabilité
- Plusieurs entreprises utilisent des logiciels de plus de 20 ans
- Les systèmes subissent des modifications de manière continue tout au long de leur existence
  - Plusieurs contributeurs, mais personne ne comprend le système en entier

Remplacement des systèmes hérités

- Remplacer un système hérité pose des **risques**

– Il n'existe généralement pas de spécification du système en entier sur laquelle se baser pour développer un nouveau système

–

– Le processus métier est souvent étroitement lié aux logiciels utilisés

– Les logiciels peuvent contenir des règles métiers non documentées

– Développer un nouveau système comporte des risques

- Retards, coûts plus élevés que prévu, etc.

Coûts des systèmes hérités

- **Les systèmes hérités deviennent plus coûteux à maintenir avec l'âge**
  - Pas de style de programmation cohérent
  - Utilisation de langages anciens
  - Documentation inadéquate ou dépassée
  - Structure corrompue par plusieurs années de maintenance ad hoc
  - Optimisation pour l'espace / temps d'exécution et non pas la compré hension du code
  - Données incohérentes, incompatibles, dupliquées, etc.

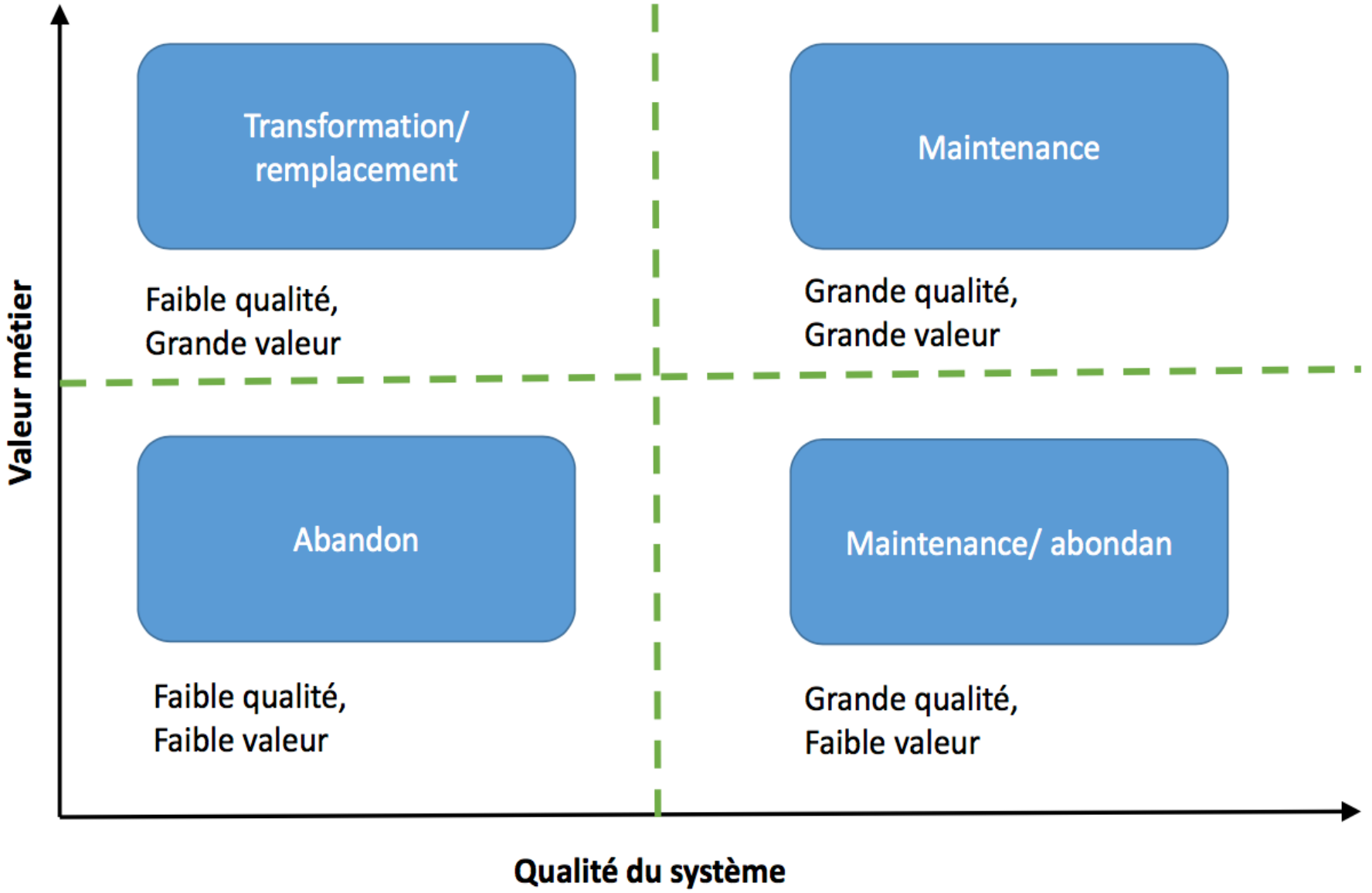
Le dilemme des systèmes hérités

- Continuer à utiliser le système accroît invariablement les coûts
- Remplacer le système coûte cher, et le nouveau système peut être moins efficace que l'ancien
- Plusieurs entreprises cherchent à allonger la durée de vie de leurs logiciels

Evaluation des systèmes hérités

- 4 options possibles
  - Abandonner le système complètement
  - Continuer à maintenir le système
  - Transformer le système pour faciliter la maintenance
  - Remplacer le système hérité par un nouveau système
- Les systèmes complexes peuvent utiliser une combinaison de plusieurs options

Evaluation des systèmes hérités



Les problèmes clés de la maintenance logicielle

- **Problèmes techniques**

– Compréhension limitée

- La rapidité avec laquelle quelqu'un peut comprendre ou faire un changement ou une correction dans un logiciel sur lequel cet individu n'a pas travaillé.

- Les études montrent que de 40 à 60% de l'effort de maintenance est consacré à la compréhension du logiciel à modifier.

- La compréhension devient encore plus difficile si elle est réalisée à partir de l'analyse directe d'un code source ou plus géné ralement lors de l'analyse de représentations orientées texte.

Les problèmes clés de la maintenance logicielle

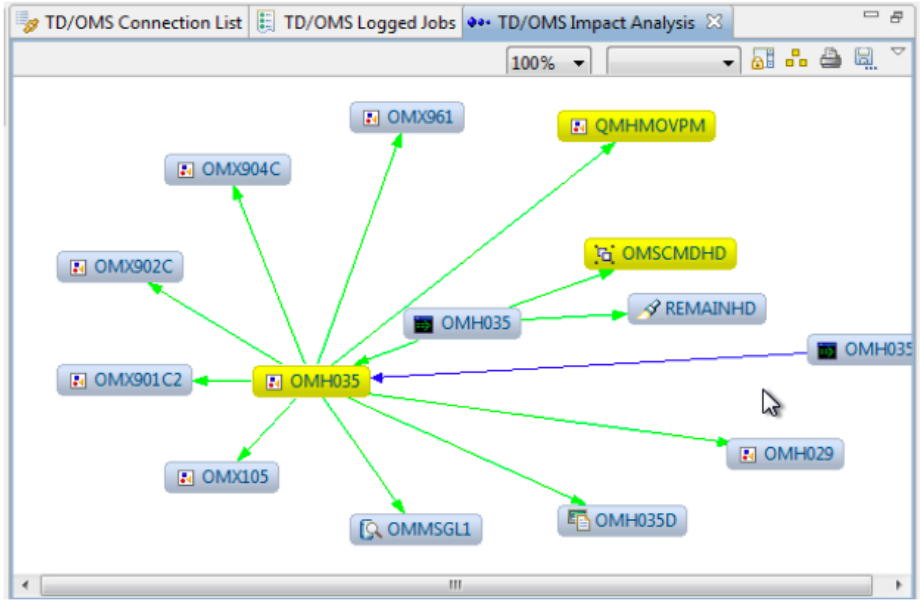
- **Problèmes techniques :**

– Test de régression :

- Tests d'un programme préalablement testé, après une modification.
- Pour s'assurer que des défauts n'ont pas été introduits ou dé couverts dans des parties non modifiées du logiciel.
- Viennent compléter les tests unitaires et les tests d'intégration en amont des tests de recette.
- Sont fastidieux, car devant être le plus exhaustifs possible, afin d'assurer que le logiciel fonctionne de la même manière.
- L'enjeu est d'identifier les tests pertinents pour minimiser l'effort de test tout en maximisant la couverture des risques de régression.

Les problèmes clés de la maintenance logicielle

- **Problèmes techniques :**
  - Analyse d'impact
- Décrit comment mener, à un coût efficace, une analyse complète des impacts d'un changement dans un logiciel existant.
- Nécessite d'analyser la structure et le contenu d'un logiciel.



Les problèmes clés de la maintenance logicielle

.Problèmes techniques :

–Analyse d'impact

.Objectifs de l'analyse d'impact :

–Détermination de la portée d'un changement pour établir un plan et implanter le travail.

–Développement d'estimations justes des ressources nécessaires pour effectuer le travail.

–L'analyse de coûts/bénéfices du changement demandé.

–Communication aux autres de la complexité d'un changement donné.

Les problèmes clés de la maintenance logicielle

- **Problèmes techniques :**
  - Maintenabilité
- Est une caractéristique liée à la qualité d'un logiciel.
- Mesure l'aisance avec laquelle un logiciel peut être maintenu, am élioré, adapté ou corrigé pour satisfaire aux exigences spécifiées [IEEE].
- Ses sous-caractéristiques doivent être spécifiées, revues et contrô lées durant les activités de développement logiciel pour réduire les coûts de maintenance.

Les problèmes clés de la maintenance logicielle

- Problèmes techniques :
  - Maintenabilité
- Les sous caractéristiques de la maintenabilité :
  - Analysabilité : mesure de l'effort du mainteneur ou des ressources utilisées pour essayer de diagnostiquer les déficiences ou les causes des pannes ou pour identifier les parties à modifier.
  -
  - Interchangeabilité : mesure de l'effort du mainteneur associé à l'implantation d'une modification spécifiée.
  -
  - Stabilité : mesure les comportements inattendus du logiciel, incluant ceux rencontrés lors des tests.
  - Testabilité : mesure de l'effort du mainteneur et de l'utilisateur pour faire des tests sur le logiciel modifié.

Les problèmes clés de la maintenance logicielle

- **Problèmes liés à la gestion de la maintenance :**

– Alignement avec les objectifs de l'entreprise

- Objectifs d'une entreprise : Retour sur investissement

– Objectif du développement logiciel : Livraison dans les délais et dans le budget définis pour remplir les besoins de l'utilisateur.

– Objectif de la maintenance :

- Allonger la vie d'un logiciel le plus longtemps possible.

- Satisfaire l'utilisateur / des mises à jour et des améliorations.

– Le retour sur investissement est moins clair par rapport au retour sur investissement du développement.

- Vue par les gestionnaires : une activité majeure consommant de grandes ressources sans bénéfices clairs et quantifiables pour l'entreprise.

Les problèmes clés de la maintenance logicielle

- **Problèmes liés à la gestion de la maintenance**
  - Constitution des équipes de maintenance
- La maintenance n'est généralement pas vue comme un travail sé duisant :
  - Par manque [de maîtrise] des fondements théoriques.
  - Par manque [de connaissances et de maîtrise] des techniques et outils.
  - ...
  - Processus
- Les étapes du processus de maintenance qui ne sont pas en commun avec le développement sont mal connues par les gestionnaires.

Les problèmes clés de la maintenance logicielle

- **Problèmes liés à la gestion de la maintenance**
  - Aspects organisationnels de la maintenance
- Décrivent comment identifier quelle organisation et/ou service sera responsable de la maintenance du logiciel.
  - Externalisation [Outsourcing]
- Les logiciels les moins critiques sont le plus souvent sujets à l'externalisation par crainte de perdre le contrôle sur les logiciels qui sont au cœur du métier de l'entreprise.
- Un des défis majeurs pour ceux qui fournissent le service externalisé est de déterminer la portée des services de maintenance requis et les détails contractuels.

Les problèmes clés de la maintenance logicielle

.Avantages de réaliser la maintenance d'un logiciel par son développeur :

- Le développeur a la meilleure connaissance du système.
- Il n'est pas nécessaire d'élaborer une documentation.
- Un système de communication formel entre les mainteneurs et les développeurs ne doit pas être établi.
- Les utilisateurs du système ne travaillent qu'avec une seule entreprise.
- Une diversité dans le travail est engendrée, satisfaisant d'autant plus le personnel de l'entreprise de développement.

Les problèmes clés de la maintenance logicielle

.Inconvénients de réaliser la maintenance d'un logiciel par son développeur :

- Le personnel peut vouloir quitter l'entreprise de développement si la maintenance devient trop lourde.
- Les nouveaux employés peuvent être mécontents suite à la quantité de travail de maintenance.
- Si les experts développeurs quittent l'entreprise, les personnes responsables de la maintenance peut ne pas être formés de manière adéquate.
- Les développeurs passent souvent trop de temps à parfaire leur système.
- Les développeurs initiaux sont souvent affectés à de nouveaux projets ou à des projets prioritaires.

Les problèmes clés de la maintenance logicielle

.Avantages de réaliser la maintenance par une entreprise extérieure :

- Une meilleure documentation est générée.
- Des procédures formelles de transition sont créées.
- Les mainteneurs connaissent les forces et les faiblesses du système.
- Des procédures d'implémentation des changements sont é tablies.

Les problèmes clés de la maintenance logicielle

.Inconvénients de réaliser la maintenance par une entreprise extérieure :

- La transition du système peut être lente ;
- Le moral peut baisser si de mauvaises personnes sont assignées au projet ;
- Il y a un réel besoin de formation ;
- Des problèmes de financement peuvent être rencontrés ;
- Prendre connaissance du système nécessite un certain temps ;
- Mettre en place l'entreprise de maintenance et ses équipements prend un certain temps ;
- Le support des utilisateurs peut en pâtir ;
- La crédibilité du support des utilisateurs peut en souffrir.

Les problèmes clés de la maintenance logicielle

.Estimation des coûts

–Les estimations des coûts de maintenance sont affectées par beaucoup de facteurs techniques et non techniques.

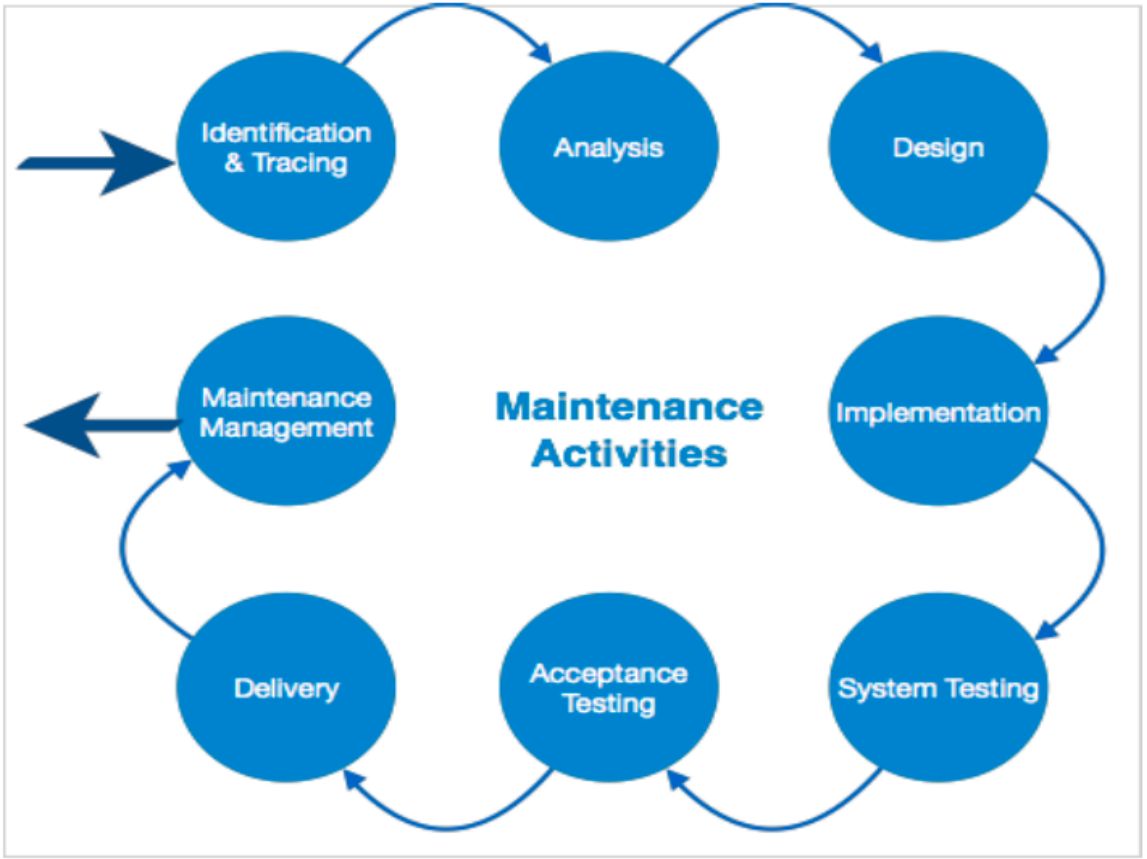
–ISO/IEC 14764 énonce que ”les deux approches les plus populaires pour l’estimation des ressources pour la maintenance de logiciel sont l’utilisation de modèles paramétrés et l’utilisation de l’expérience”.

–Les modèles paramétrés se basent sur les données des projets et discutent de tous les aspects de l’estimation des coûts, incluant les points de fonction.

–Expérience : par le jugement d’experts.

Processus de maintenance

- En 1993, l'IEEE a publié un processus itératif standard pour la gestion et l'exécution des activités de maintenance logicielle sous le nom de "IEEE Standard for Software Maintenance".



Processus de maintenance

.Classification et Identification

- Le processus démarre par la soumission d'une requête de modification.
- L'entreprise de maintenance détermine tout d'abord le type de requête dont il s'agit (corrective, perfective ou adaptative).
- Une priorité est ensuite assignée à cette requête ainsi qu'un numéro unique.
- Toutes ces informations sont entrées automatiquement dans une base de données qui pourra être consultée tout au long du cycle.

Processus de maintenance

.Analyse

–Une étude de faisabilité ainsi qu'une analyse détaillée sont effectuées durant cette phase.

–Le mainteneur prêtera attention aux impacts des modifications.

–Il proposera plusieurs solutions alternatives et définira les coûts.

–L'analyse détaillée fournira différentes informations telles que des plans d'implémentation, des stratégies de test ou encore des identifications de fonctionnalités.

–A la fin de cette phase d'analyse, l'ensemble des changements doit être totalement planifié.

Processus de maintenance

- **Conception**
  - Durant la phase de conception, l'ensemble des informations collectées pendant la phase précédente sont rassemblées et utilisées afin de construire le design de l'application.
  - En fin de conception, la documentation du logiciel doit être mise à jour (plans de tests, design, analyse et exigences).
- **Implémentation**
  - La phase d'implémentation comprend l'ensemble des processus de codage, de tests unitaires, d'intégration, d'analyse des risques et de revue de code.
  - Toutes les documentations concernant le logiciel sont mises à jour.

Processus de maintenance

.Test Système

–La phase des tests du système s'assure que l'ensemble des exigences originelles de l'utilisateur sont toujours rencontrées de même que les nouvelles ajoutées.

–La phase de tests est l'une des plus importante du cycle car elle permet de s'assurer que le nouveau produit est toujours satisfaisant et que les nouveaux changements ont été implémentés Correctement.

Processus de maintenance

.Test d'acceptation

–S'établit sur un système totalement intégré et est réalisée par les clients, utilisateurs ou par un tiers parti.

–Les testeurs ont pour objectif de rapporter les erreurs et de dé terminer si les fonctionnalités du système sont en accord avec leurs exigences.

.Livraison

–Le fournisseur livre le nouveau système aux utilisateurs. Il effectue la configuration du système et la formation des utilisateurs. Après cette phase, le système est prêt à fonctionner de manière autonome.

Techniques pour la maintenance

- La compréhension du logiciel est «le processus qui vise à comprendre le code source d'un logiciel "(Deimel & Naveda, 1990).

– La tâche de construire des modèles mentaux du logiciel sous-jacent à différents niveaux d'abstraction, allant de modèles du code lui-même, à des modèles du domaine de l'application sous-jacente...".

– Les approches pour la compréhension

- Approche ascendante (Botton-Up)

– Regrouper ensemble des éléments du code source pour construire des éléments plus abstraits.

- Approche descendante (Top-down)

– Utiliser les connaissances concernant le domaine du logiciel pour établir des correspondances avec le code source.

Techniques pour la maintenance

- **Réingénierie**
  - La restructuration ou la réécriture de tout ou partie d'un système sans changer sa fonctionnalité.
  - Applicable lorsque certains (mais pas tous) sous-systèmes d'un système plus large demandent d'être maintenus fréquemment.
  - Par exemple
- Unix a été développé dans le langage assembleur et il a été re-conçu en C.

Techniques pour la maintenance

.Processus de réingénierie

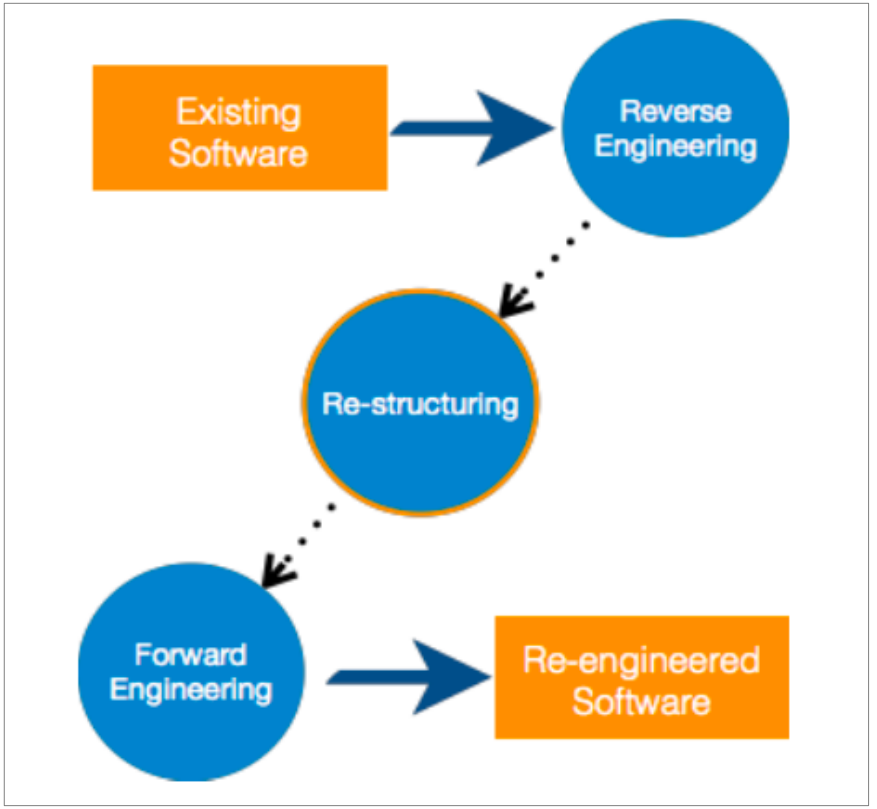
–Décider par rapport à la portée de la réing énierie : tout ou une partie du logiciel ?

–Appliquer la rétro-ingénierie, afin d'obtenir les spécifications des logiciels existants.

–Restructurer le logiciel si nécessaire. Par exemple, transformer un code procédural en un code orienté objet.

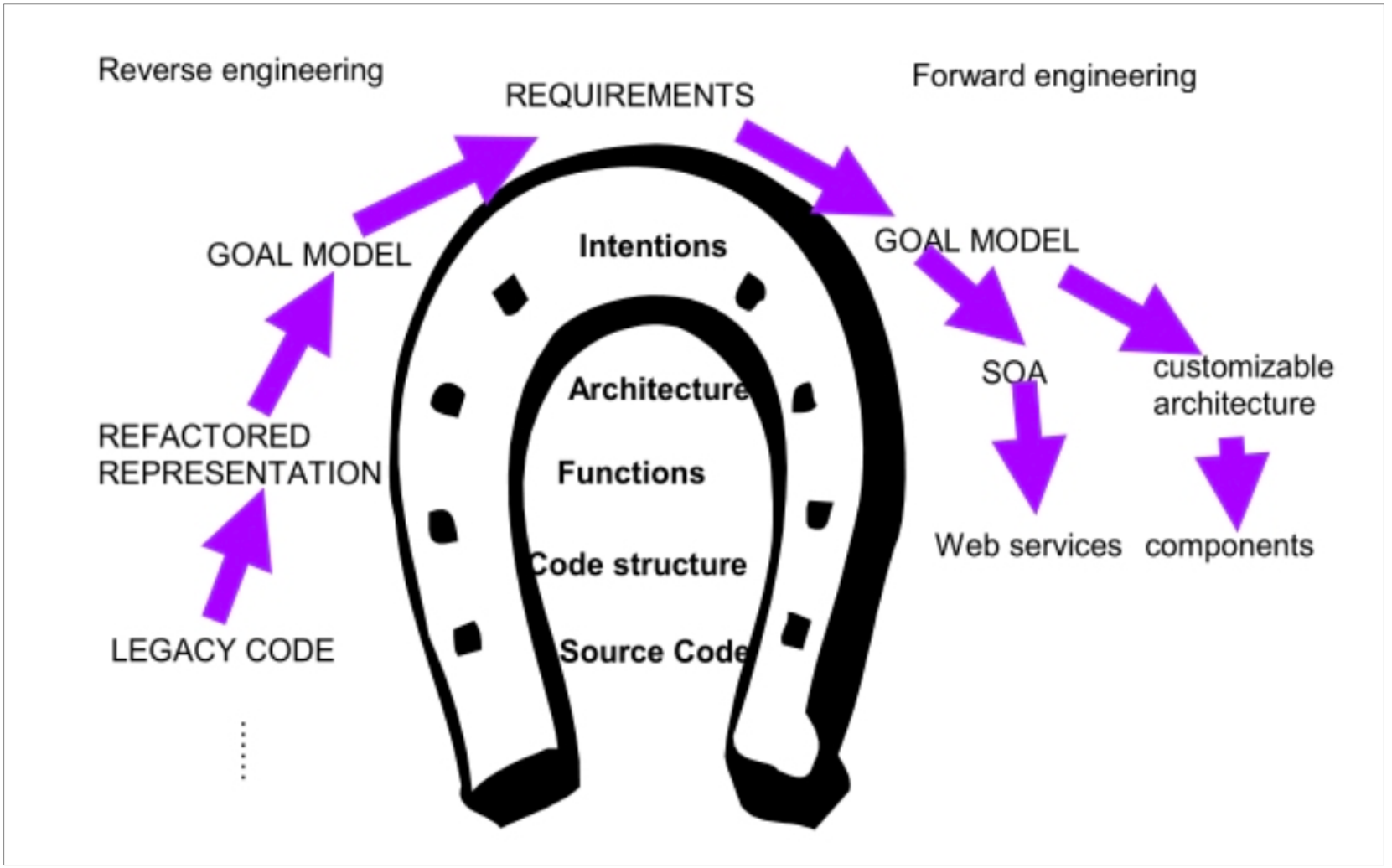
–Restructurer les données si nécessaires

–Appliquer les concepts d'ingénierie afin d'obtenir le logiciel après sa réingénierie.



Techniques pour la maintenance

.Processus de réingénierie



Techniques pour la maintenance

- **La rétro-ingénierie logicielle**

  - Analyser un programme, sans le bénéfice des spécification originales, pour en comprendre le fonctionnement interne p la création d'une représentation à un plus haut nive d'abstraction.

  - Deux approches complémentaires sont en général utilisé :

- Analyse statique : Analyser le code d'un logiciel pour obte des informations sur son comportement lors de son e cution sans réellement l'exécuter.

- Analyse dynamique : Analyser le code d'un logiciel po obtenir des informations sur son comportement lors de s exécution en l'exécutant avec des entrées suffisamment v ées.

Source : http://www.tutorialspoint.com/

Techniques pour la maintenance

- SANER 2015 :22nd IEEE Inter. Conf. on Software Analysis, Evolution, and Reengineering
  - Empirical studies in reverse engineering
  - Binary reverse engineering
  - Program analysis and slicing
  - Re-documenting legacy systems
  - Reengineering patterns
  - Model-driven reengineering
  - Program transformation and refactoring
  - Reverse engineering of service-oriented and legacy systems
  - User interface reengineering
  - Reengineering to distributed and cloud systems
  - Mining software repositories for software analysis
  - Software architecture recovery
  - Visualization techniques and tools
  - Program Comprehension

Techniques pour la maintenance

- SANER 2015 :22nd IEEE Inter. Conf. on Software Analysis, Evolution, and Reengineering

  - Concept, feature and service location
  - Object and aspect identification
  - Dynamic analysis
  - Wrapping techniques
  - Data reverse engineering
  - Preprocessing, parsing and fact extraction ●Reverse engineering for security assessment
  - Reverse engineering tool support
  - Education in reverse engineering
  - erformance re-engineering
  - Software quality
  - Search-based software engineering ●Evolution in/of the cloud