

Gestion de projets informatiques complexes



une brève histoire du management
des systèmes complexes en
environnement distribué,
racontée aux petits enfants.

*«Tout problème simple a une solution
complexe...
... qui ne fonctionne pas».*
O.Lockert

© 2003-2018 CARAPACE
Antoine CELIER

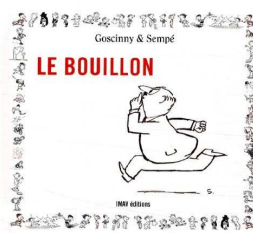


Projets informatiques

MODELE DE COMPOSANTS

Le Bouillon

Référence : CARAPACE-158 Version 10/28/2018



Diviser pour mieux régner

Designer versus Architecte

L'architecture Logique

- ❑ Objectifs de l'Architecture Logique
- ❑ Critère de Qualité Logicielle
- ❑ Réduire la dégradation du Logiciel
- ❑ Développer pour et par la Réutilisation

Les modèles d'architecture

- ❑ Le modèle conventionnel
- ❑ Le modèle des 4+1 vues

Les styles architecturaux

Construire son Architecture Logique

Facteurs influençant l'Architecture Logique

- ❑ Respect du secret médical
- ❑ NLS
- ❑ Gestion des erreurs
- ❑ Encapsulation
- ❑ La Gestion dynamique des menus

Où en sommes-nous

MODELE DE COMPOSANTS

Session 158 - page 2

© 2003-2018 CARAPACE

Gestion de projets informatique complexes



1

Designer versus Architecte

- **Le Viaduc de Millau est enseigné dans les écoles de management comme un modèle de Gestion de Projet**
 - 13 ans d'études techniques et financières - 3 ans de réalisation



MODELE DE COMPOSANTS
Session 158 - page 3

© 2003-2018 CARAPACE



Gestion de projets informatique complexes

1

Designer versus Architecte

- **Plusieurs points permettent de mieux comprendre comment ce programme a été géré :**

- **Rigueur** : la qualité du travail d'Architecture de Michel Virlogeux : un travail très long et très soigné de structuration et de planification du chantier
- **Bon sens** : le savoir-faire en Design de l'architecte Norman Foster
- **Communication** : les qualités humaines de Marc Buonomo, chef de chantier



MODELE DE COMPOSANTS
Session 158 - page 4

© 2003-2018 CARAPACE



Gestion de projets informatique complexes

1

Designer versus Architecte

- **Le Designer décrit le « quoi faire »**
 - Ce que doit réaliser un système informatique



MODELE DE COMPOSANTS
Session 158 - page 5

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



1

Designer versus Architecte

- **L'Architecte décrit le « comment le faire »**
 - Comment le système informatique doit être conçu de manière à répondre aux spécifications



MODELE DE COMPOSANTS
Session 158 - page 6

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



2

L'Architecture Logique

■ L'architecture logique

- Décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions
- Constitue le plus gros livrable d'un processus logiciel après le produit (le logiciel lui-même).
- On considère que le travail de conception est au moins égale à celui du codage de l'application

2.1

Objectifs de l'Architecture Logique

■ Les deux objectifs principaux de toute architecture logique sont :

- la réduction des coûts
- la qualité du logiciel

■ La réduction des coûts est principalement réalisée par:

- La réutilisation de composants logiciels
- La diminution du temps de maintenance

■ La qualité, par contre, se trouve distribuée à travers plusieurs critères

- La norme ISO 9126 est un exemple d'un tel ensemble de critères.

2.1

Objectifs de l'Architecture Logique

- **Capacité fonctionnelle : est-ce que le logiciel répond aux besoins fonctionnels exprimés ?**
 - Pertinence – Exactitude – Interopérabilité
 - Sécurité – Conformité
- **Fiabilité : est-ce que le logiciel maintient son niveau de service dans des conditions précises et pendant une période déterminée ?**
 - Maturité (faible fréquence d'apparition des incidents)
 - Tolérance aux pannes
 - Facilité de récupération : capacité d'un logiciel défectueux à retourner dans un état opérationnel complet (données et connexions réseaux incluses)

2.1

Objectifs de l'Architecture Logique

- **Facilité d'utilisation : est-ce que le logiciel requiert peu d'effort à l'utilisation ?**
 - Facilité de compréhension
 - Facilité d'apprentissage
 - Facilité d'exploitation
- **Rendement et efficacité : est-ce que le logiciel requiert un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences ?**
 - Comportement temporel : temps de réponse, taux de transactions
 - Utilisation des ressources : mémoire, processeur, disque et réseau

2.1

Objectifs de l'Architecture Logique

- **Maintenabilité : est-ce que le logiciel requiert peu d'effort à son évolution par rapport aux nouveaux besoins ?**
 - Facilité d'analyse : identification dans le logiciel de l'origine d'un défaut constaté
 - Facilité de modification
 - Stabilité - Testabilité
- **Portabilité : est-ce que le logiciel peut être transféré d'une plate-forme ou d'un environnement à un autre ?**
 - Facilité d'adaptation à des changements de spécifications ou d'environnements opérationnels
 - Facilité d'installation - Coexistence
 - Interchangeabilité : utilisation de greffons

2.2

Critères de Qualité Logicielle

- **L'interopérabilité extrinsèque**
 - Exprime la capacité du logiciel à communiquer et à utiliser les ressources d'autres logiciels comme les documents créés par une certaine application.
- **L'interopérabilité intrinsèque**
 - Exprime le degré de cohérence entre le fonctionnement des commandes et des modules à l'intérieur d'un système ou d'un logiciel.
- **La portabilité**
 - Exprime la possibilité de compiler le code source et/ou d'exécuter le logiciel sur des plates-formes (machines, systèmes d'exploitation, environnements) différentes.

2.2

Critères de Qualité Logicielle

- **La compatibilité**
 - Exprime la possibilité, pour un logiciel, de fonctionner correctement dans un environnement ancien (compatibilité descendante) ou plus récent (compatibilité ascendante).
- **La validité**
 - Exprime la conformité des fonctionnalités du logiciel avec celles décrites dans le cahier des charges.
- **La vérifiabilité**
 - Exprime la simplicité de vérification de la validité.
- **L'intégrité**
 - Exprime la faculté du logiciel à protéger ses fonctions et ses données d'accès non autorisés.

2.2

Critères de Qualité Logicielle

- **La fiabilité**
 - Exprime la faculté du logiciel à gérer correctement ses propres erreurs de fonctionnement en cours d'exécution.
- **La maintenabilité**
 - Exprime la simplicité de correction et de modification du logiciel, et même, parfois, la possibilité de modification du logiciel en cours d'exécution.
- **La réutilisabilité**
 - Exprime la capacité de concevoir le logiciel avec des composants déjà conçus tout en permettant la réutilisation simple de ses propres composants pour le développement d'autres logiciels.

2.2

Critères de Qualité Logicielle

- **L'extensibilité**
 - Exprime la possibilité d'étendre simplement les fonctionnalités d'un logiciel sans compromettre son intégrité et sa fiabilité.
- **L'efficacité**
 - Exprime la capacité du logiciel à exploiter au mieux les ressources offertes par la ou les machines où le logiciel sera implanté.
- **L'autonomie**
 - Exprime la capacité de contrôle de son exécution, de ses données et de ses communications.
- **La transparence E**
 - Exprime la capacité pour un logiciel de masquer à l'utilisateur (humain ou machine) les détails inutiles à l'utilisation de ses fonctionnalités.

2.2

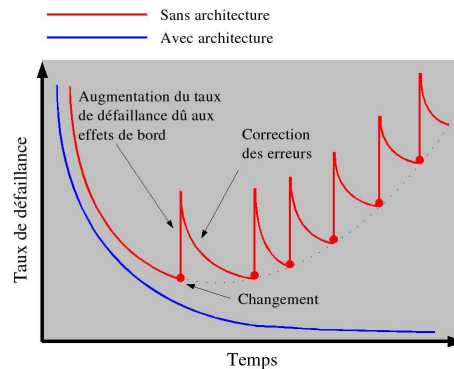
Critères de Qualité Logicielle

- **La composabilité**
 - Exprime la capacité pour un logiciel de combiner des informations provenant de sources différentes.
- **La convivialité**
 - Décrit la facilité d'apprentissage et d'utilisation du logiciel par les usagers.

2.3

Réduire la dégradation du Logiciel

- Une architecture faible ou absente peut entraîner de graves problèmes lors de la maintenance du logiciel.
 - En effet, toute modification d'un logiciel mal architecturé peut déstabiliser la structure de celui-ci et entraîner, à la longue, une dégradation (principe d'entropie du logiciel).
 - L'architecte informatique devrait donc concevoir, systématiquement, une architecture maintenable et extensible.



MODELE DE COMPOSANTS
Session 158 - page 17

Gestion de projets informatique complexes

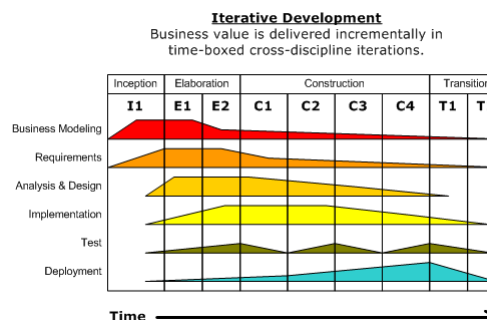
© 2003-2018 CARAPACE



2.3

Réduire la dégradation du Logiciel

- Dans les processus itératifs comme UP (Unified Process), la gestion des changements est primordiale.
 - En effet, il est implicitement considéré que les besoins des utilisateurs du système peuvent changer et que l'environnement du système peut changer.
 - L'architecte informatique a donc la responsabilité de prévoir le pire et de concevoir l'architecture en conséquence : la plus maintenable possible et la plus extensible possible.



MODELE DE COMPOSANTS
Session 158 - page 18

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



2.3

Réduire la dégradation du Logiciel

■ Bien des logiciels

- Ont été créés sans architecture par plusieurs générations de développeurs, ayant chacun usé d'une imagination débordante pour réussir à maintenir l'intégrité du système.
- Une telle absence d'architecture peut être qualifiée **d'architecture organique**. En effet, un développeur confronté à une telle architecture a plus l'impression de travailler avec un organisme vivant qu'avec un produit industriel.
- Il en résulte que la complexité du logiciel fait en sorte que celui-ci est extrêmement difficile à comprendre et à modifier.
- À la limite, modifier une partie du système est plus proche, en complexité, de la transplantation cardiaque que du changement de carburateur

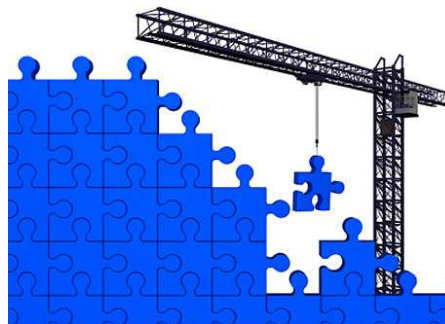


2.4

Develop. pour et par la réutilisation

■ La réutilisation de composants logiciels

- Est l'activité permettant de réaliser les économies les plus substantielles
- Encore faut-il posséder des composants à réutiliser.
- De plus, la réutilisation de composants nécessite de créer une architecture logicielle permettant une intégration harmonieuse de ces composants.
- Le développement par la réutilisation logicielle impose donc un cycle de production-réutilisation perpétuel et une architecture logicielle normalisée.



2.4

Develop. pour et par la réutilisation

■ Framework

- Une réutilisation bien orchestrée nécessite la création et le maintien d'une bibliothèque logicielle et un changement de focus; créer une application revient à créer les composants de bibliothèque nécessaires puis à construire l'application à l'aide de ces composants.
- Une telle bibliothèque, facilitant le développement d'application est un Framework d'entreprise et son architecture, ainsi que sa documentation sont les pierres angulaires de la réutilisation logicielle en entreprise.



2.4

Develop. pour et par la réutilisation

■ Rôle de l'Architecte

- Le rôle de l'architecte informatique se déplace donc vers celui de bibliothécaire.
- L'architecte informatique doit explorer la bibliothèque pour trouver les composants logiciels appropriés puis créer les composants manquants, les documenter et les intégrer à la bibliothèque.
- Dans une grande entreprise, ce rôle de bibliothécaire est rempli par l'architecte informatique en chef qui est responsable du développement harmonieux de la bibliothèque et de la conservation de l'intégrité de son architecture.



3

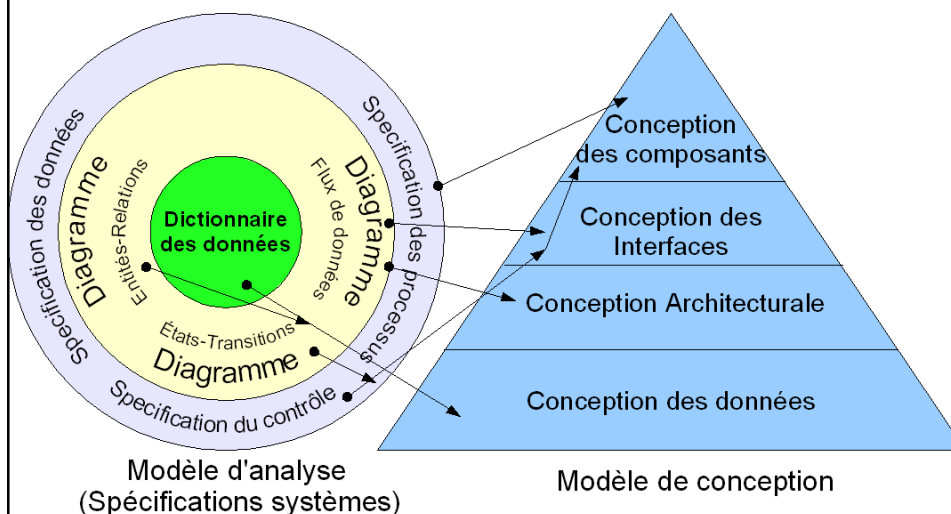
Les modèles d'architecture

■ La description d'un système complexe

- Tel un logiciel informatique peut être faite selon plusieurs points de vue différents mais chacun obéit à la formule de Perry et Wolf:
architecture = éléments + formes + motivations
- Selon le niveau de granularité, les éléments peuvent varier en tailles (lignes de code, procédures ou fonctions, modules ou classes, paquetages ou couches, applications ou systèmes informatiques), ils peuvent varier en raffinement (ébauche, solution à améliorer ou solution finale) et en abstraction (idées ou concepts, classes ou objets, composants logiciels).
- Les éléments peuvent également posséder une *temporalité* (une existence limitée dans le temps) et une *localisation* (une existence limitée dans l'espace).

3.1

Le modèle conventionnel



3.1

Le modèle conventionnel

■ Le modèle conventionnel

- ❑ Ce diagramme décrit, à gauche, les spécifications systèmes qui sont également représentées par des diagrammes (Entités-Relations, Flux de données, États-Transitions).
- ❑ Et à droite, nous avons les différentes activités de conception prenant comme intrants les livrables de la phase d'analyse.
- ❑ Nous voyons que l'architecture logicielle traditionnelle nécessiterait de produire au moins quatre vues distinctes : une architecture des données (conception des données), une architecture fonctionnelle et/ou modulaire (conception architecturale), une autre architecture fonctionnelle et/ou modulaire pour les interfaces utilisateurs (conception des interfaces) et une architecture détaillée (ordinogrammes, états-transitions) des différents modules (conception des composants).



3.1

Le modèle conventionnel

■ La pyramide

- ❑ Exprime que chaque couche est bâtie sur la précédente.
- ❑ En effet, les composants réalisant les fonctionnalités du logiciel doivent manipuler des éléments de données qui doivent donc être préalablement décrits.
- ❑ De même, les composants réalisant les interfaces utilisateurs doivent utiliser les fonctionnalités du logiciel préalablement décrites.
- ❑ Et finalement, la création de l'architecture détaillée de chacun des composants du logiciel nécessite, évidemment, que ceux-ci soient préalablement inventés.

■ Ce modèle d'architecture impose une séparation claire entre les données, les traitements et la présentation.



3.1

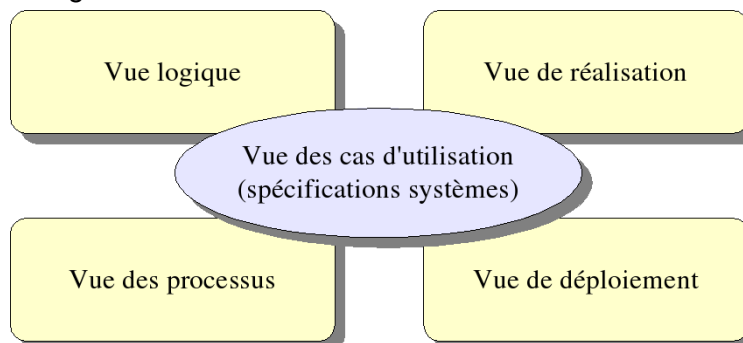
Le modèle conventionnel

- **Modèle d'analyse ou modèle d'architecture ?**
 - Puisque l'analyse produit également des diagrammes, il est naturel de se questionner
 - En effet, quand se termine l'analyse et quand commence l'architecture ?
 - La réponse à cette question est fort simple
 - Les éléments des diagrammes d'analyse correspondent à des éléments visibles et compréhensibles par les utilisateurs du système
 - Alors que les éléments des diagrammes d'architectures ne correspondent à aucune réalité tangible pour ceux-ci.

3.2

Le modèle des 4 + 1 vues

- **Le modèle de Kruchten dit modèle des 4 + 1 vues est celui adopté dans l'Unified Process (*)**
 - Ici encore, le modèle d'analyse, baptisé vue des cas d'utilisation, constitue le lien et motive la création de tous les diagrammes d'architecture.



(*) PU vient compléter la systématique des modèles UML

3.2

Le modèle des 4 + 1 vues

■ La vue des cas d'utilisation

- La vue des cas d'utilisation est un modèle d'analyse formalisé par Ivar Jacobson.
- Un cas d'utilisation est défini comme un ensemble de scénarios d'utilisation, chaque scénario représentant une séquence d'interaction des utilisateurs (acteurs) avec le système.
- L'intérêt des cas d'utilisation est de piloter l'analyse par les exigences des utilisateurs. Ceux-ci se sentent concernés car ils peuvent facilement comprendre les cas d'utilisation qui les concernent.

3.2

Le modèle des 4 + 1 vues

■ La vue des cas d'utilisation

- Cette méthode permet donc d'aider à formaliser les véritables besoins et attentes des utilisateurs; leurs critiques et commentaires étant les briques de la spécification du système.
- L'ensemble des cas d'utilisation du logiciel en cours de spécification est représenté par un diagramme de cas d'utilisation, chacun des scénarios de celui-ci étant décrit par un ou plusieurs diagrammes dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.

3.2

Le modèle des 4 + 1 vues

■ La vue logique

- La vue logique constitue la principale description architecturale d'un système informatique et beaucoup de petits projets se contentent de cette seule vue.
- Cette vue décrit, de façon statique et dynamique, le système en termes d'objets et de classes.
- La vue logique permet d'identifier les différents éléments et mécanismes du système à réaliser.
- Elle permet de décomposer le système en abstractions et constitue le coeur de la réutilisation. En effet, l'architecte informatique récupérera un maximum de composants des différentes bibliothèques et cadres (framework) à sa disposition. Une recherche active de composants libres et/ou commerciaux pourra également être envisagée.

3.2

Le modèle des 4 + 1 vues

■ La vue logique

- La vue logique est représentée, principalement, par des diagrammes statiques de classes et d'objets enrichis de descriptions dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.

3.2

Le modèle des 4 + 1 vues

■ La vue des processus

- ❑ La vue des processus décrit les interactions entre les différents processus, threads (fils d'exécution) ou tâches, elle permet également d'exprimer la synchronisation et l'allocation des objets.
- ❑ Cette vue permet avant tout de vérifier le respect des contraintes de fiabilité, d'efficacité et de performances des systèmes multitâches.
- ❑ Les diagrammes utilisés dans la vue des processus sont exclusivement dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions.



3.2

Le modèle des 4 + 1 vues

■ La vue de réalisation

- ❑ La vue de réalisation permet de visualiser l'organisation des composants (bibliothèque dynamique et statique, code source...) dans l'environnement de développement.
- ❑ Elle permet aux développeurs de se retrouver dans le capharnaüm que peut être un projet de développement informatique. Cette vue permet également de gérer la configuration (auteurs, versions...).
- ❑ Les seuls diagrammes de cette vue sont les diagrammes de composants.



3.2

Le modèle des 4 + 1 vues

■ La vue de déploiement

- La vue de déploiement représente le système dans son environnement d'exécution.
- Elle traite des contraintes géographiques (distribution des processeurs dans l'espace), des contraintes de bandes passantes, du temps de réponse et des performances du système ainsi que de la tolérance aux fautes et aux pannes.
- Cette vue est fort utile pour l'installation et la maintenance régulière du système.
- Les diagrammes de cette vue sont les diagrammes de composants et les diagrammes de déploiement.



4

Les styles architecturaux

■ Les styles architecturaux

- L'architecture logicielle, tout comme l'architecture traditionnelle, peut se catégoriser en styles
- En effet, malgré les millions de systèmes informatiques construits de par le monde au cours des cinquante dernières années, tous se classent parmi un nombre extrêmement restreint de styles architecturaux
- De plus, un système informatique peut utiliser plusieurs styles selon le niveau de granularité ou l'aspect du système décrit.
- Nous ferons remarquer que, comme en architecture traditionnelle, c'est souvent par le mélange d'anciens styles que les nouveaux apparaissent.



4.1 Architecture en appels et retours

■ Architecture en appels et retours

- L'architecture en appels et retours est basée sur le raffinement graduel proposé par Niklaus Wirth.
- Cette approche, également appelée décomposition fonctionnelle, consiste à découper une fonctionnalité en sous-fonctionnalités qui sont également divisées en sous-sous-fonctionnalités et ainsi de suite.
- La devise diviser pour régner est souvent utilisée pour décrire cette démarche.

4.1 Architecture en appels et retours

■ Architecture en appels et retours

- Si à l'origine cette architecture était fondée sur l'utilisation de fonctions, le passage à une méthode modulaire ou objet est toute naturelle
- La fonctionnalité d'un module ou d'un objet est réalisée par des sous-modules ou des sous-objets baptisés travailleurs (worker).
- Le terme hiérarchie de contrôle est alors utilisé pour décrire l'extension de cette architecture au paradigme modulaire ou objet.
- Une forme dérivée de cette architecture est l'architecture distribuée où les fonctions, modules ou classes se retrouvent répartis sur un réseau.

4.2

Architecture en couches

■ Architecture en couches

- La conception de logiciels nécessite de recourir à des bibliothèques.
- Une bibliothèque très spécialisée utilise des bibliothèques moins spécialisées qui elles-mêmes utilisent des bibliothèques génériques.
- De plus, comme nous l'avons déjà mentionné, le développement efficace de composants réutilisables nécessite de créer une bibliothèque logicielle.



4.2

Architecture en couches

■ Architecture en couches

- L'architecture en couches est la conséquence inéluctable d'une telle approche.
- En effet, les nouveaux composants utilisent les anciens et ainsi de suite, la bibliothèque tend donc à devenir une sorte d'empilement de composants.
- La division en couches consiste alors à regrouper les composants possédant une grande cohésion (sémantiques semblables) de manière à créer un empilement de paquets de composants
- Tous les composants des couches supérieures dépendent fonctionnellement des composants des couches inférieures.



4.3

Architecture centrée sur les données

■ Architecture centrée sur les données

- Dans l'architecture centrée sur les données, un composant central (SGBD, Datawarehouse, Blackboard) est responsable de la gestion des données (conservation, ajout, retrait, mise à jour, synchronisation, ...) .
- Les composants périphériques, baptisés clients, utilisent le composant central, baptisé serveur de données, qui se comporte, en général, de façon passive (SGBD, Datawarehouse).
- Un serveur passif ne fait qu'obéir aveuglément aux ordres alors qu'un serveur actif (Blackboard) peut notifier un client si un changement aux données qui le concerne se produit.

4.3

Architecture centrée sur les données

■ Architecture centrée sur les données

- Cette architecture sépare clairement les données (serveurs) des traitements et de la présentation (clients) et permet ainsi une très grande intégrabilité, en effet, des clients peuvent être ajoutés sans affecter les autres clients.
- Par contre, tous les clients sont dépendants de l'architecture des données qui doit rester stable et qui est donc peu extensible.
- Ce style nécessite donc un investissement très important dans l'architecture des données.
- Les datawarehouses et les bases de données fédérées sont des extensions de cette architecture.

4.4

Architecture en flot de données

■ Architecture en flot de données

- ❑ L'architecture en flot de données est composée de plusieurs composants logiciels reliés entre eux par des flux de données.
- ❑ L'information circule dans le réseau et est transformée par les différents composants qu'elle traverse.
- ❑ Lorsque les composants se distribuent sur une seule ligne et qu'ils ne font que passer l'information transformée à leur voisin, on parle alors d'architecture par lot (batch).
- ❑ Si les composants sont répartis sur un réseau informatique et qu'ils réalisent des transformations et des synthèses intelligentes de l'information, on parle alors d'architecture de médiation. Les architectures orientées événements font également partie de cette catégorie.

4.5

Architecture orientée objets

■ Architecture orientée objets

- ❑ Les composants du système (objets) intègrent des données et les opérations de traitement de ces données.
- ❑ La communication et la coordination entre les objets sont réalisées par un mécanisme de passage de messages.
- ❑ L'architecture orientée objets est souvent décrite par les trois piliers : encapsulation, héritage et polymorphisme.
- ❑ L'encapsulation concerne l'architecture détaillée de chaque objet, les données étant protégées d'accès direct par une couche d'interface.
- ❑ De plus, les sous-fonctions, inutiles pour utiliser l'objet, sont masquées à l'utilisateur de l'objet.

4.5

Architecture orientée objets

■ Architecture orientée objets

- L'héritage permet d'éviter la redondance de code et facilite l'extensibilité du logiciel, les fonctionnalités communes à plusieurs classes d'objets étant regroupées dans un ancêtre commun.
- Le polymorphisme permet d'utiliser des objets différents (possédant des comportements distincts) de manière identique, cette possibilité est réalisée par la définition d'interfaces à implémenter (classes abstraites).



4.6

Architecture orientée agents

■ Architecture orientée agents

- L'architecture orientée agents correspond à un paradigme où l'objet, de composant passif, devient un composant projectif :
- En effet, dans la conception objet, l'objet est essentiellement un composant passif, offrant des services, et utilisant d'autres objets pour réaliser ses fonctionnalités
- L'architecture objet n'est donc qu'une extension de l'architecture en appels et retours, le programme peut être écrit de manière à demeurer déterministe et prédictible.



4.6

Architecture orientée agents

■ Architecture orientée agents

- L'agent logiciel, par contre, utilise de manière relativement autonome, avec une capacité d'exécution propre, les autres agents pour réaliser ses objectifs
- Il établit des dialogues avec les autres agents, il négocie et échange de l'information, décide à chaque instant avec quels agents communiquer en fonction de ses besoins immédiats et des disponibilités des autres agents.

MODELE DE COMPOSANTS

Session 158 - page 47

Gestion de projets informatiques complexes

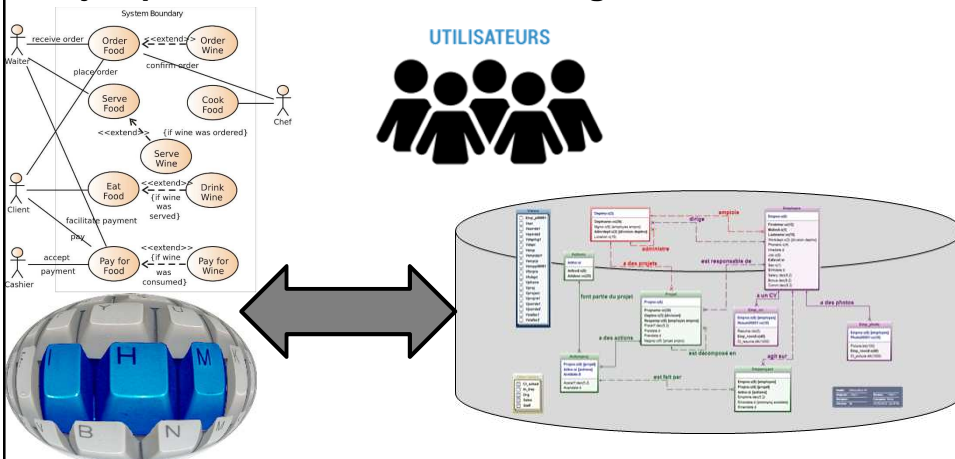
© 2003-2018 CARAPACE



5

Construire son Architecture Logique

- L'A. L., c'est le processus de décomposition de l'application, depuis les Use Cases et les règles d'IHM, jusqu'à la couche de Data-Management



MODELE DE COMPOSANTS

Session 158 - page 48

Gestion de projets informatiques complexes

© 2003-2018 CARAPACE

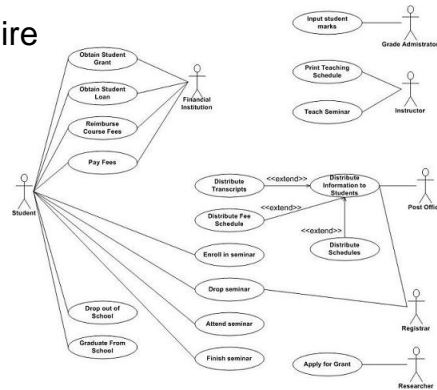


5.1

Partons des Use Cases

■ On peut partir des Use Cases de la version V1

- Médecin
- Infirmier
- Secrétaire médicale
- Responsable de laboratoire
- Data Manager



MODELE DE COMPOSANTS

Session 158 - page 49

Gestion de projets informatiques complexes

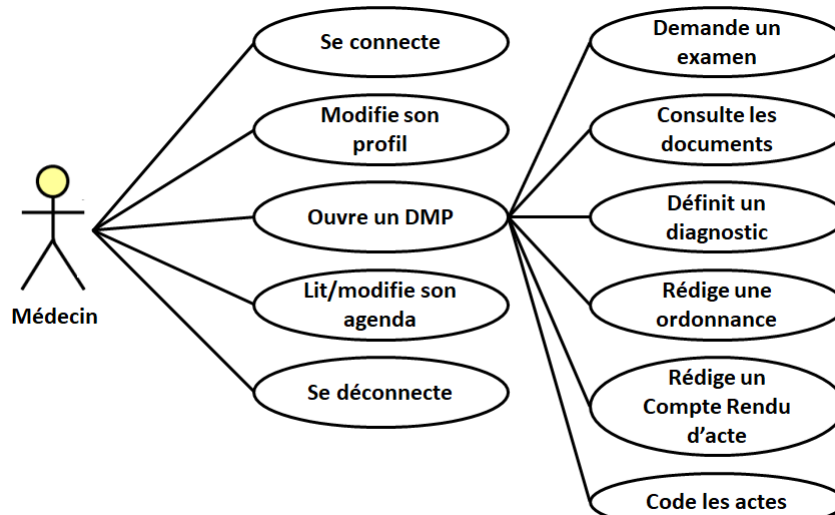
© 2003-2018 CARAPACE



5.1

Partons des Use Cases

■ On peut partir du médecin



MODELE DE COMPOSANTS

Session 158 - page 50

Gestion de projets informatiques complexes

© 2003-2018 CARAPACE



5.1

Partons des Use Cases

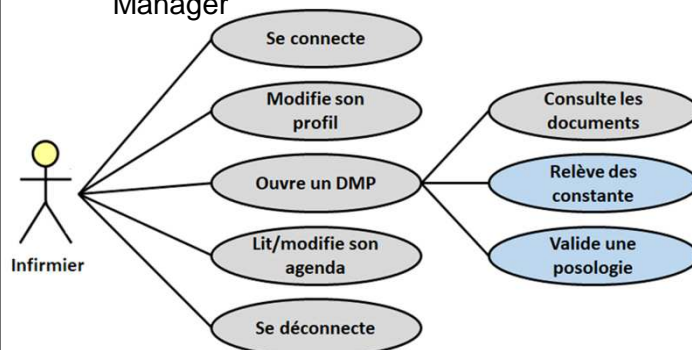
- On imagine de créer un composant Médecin
 - Avec 15 commandes

Composant MEDECIN	
Cmd1	Se connecter
Cmd2	Se déconnecter
Cmd3	Modifier son profil
Cmd4	Lire/Modifier son agenda
Cmd5	Ouvrir un DMP
Cmd6	Demander un examen
Cmd7	Consulter les documents
Cmd8	Définir un Diagnostic
Cmd9	Publier un Diagnostic
Cmd10	Rédiger une ordonnance
Cmd11	Publier une ordonnance
Cmd12	Rédiger un compte rendu d'acte
Cmd13	Publier un acte
Cmd14	Coder des actes
Cmd15	Publier un codage

5.1

Partons des Use Cases

- Mais en regardant le Use Case de l'infirmier
 - On s'aperçoit que de nombreuses commande sont communes
 - Et qu'il en serait de même pour la secrétaire médicale, du responsable de laboratoire, et même en partie du Data Manager



5.1

Partons des Use Cases

- On va donc créer des composants standards

Composant CONNEXION	
Cmd1	Se connecter
Cmd2	Se déconnecter

Composant AGENDA	
Cmd1	Lire/Modifier un agenda

Composant PROFIL	
Cmd1	Modifier un profil

Composant DMP	
Cmd1	Ouvrir un DMP
Cmd2	Consulter les documents

Composant MEDECIN	
Cmd1	Demander un examen
Cmd2	Définir un Diagnostic
Cmd3	Publier un Diagnostic
Cmd4	Rédiger une ordonnance
Cmd5	Publier une ordonnance
Cmd6	Rédiger un compte rendu d'acte
Cmd7	Publier un acte
Cmd8	Coder des actes
Cmd9	Publier un codage

MODELE DE COMPOSANTS
Session 158 - page 53

© 2003-2018 CARAPACE






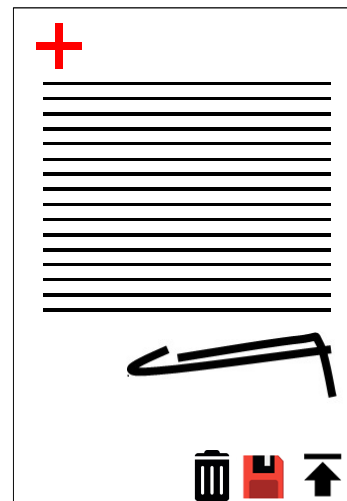
Gestion de projets informatique complexes

5.1

Partons des Use Cases

- Allons plus loin

- Il n'y a pas de différence entre publier un diagnostic, une ordonnance, un acte ou un codage
- Lors de l'édition d'un document (diagnostic, ordonnance...) on doit proposer :
 - Poubelliser 
 - Sauvegarder en brouillon 
 - Publier 



MODELE DE COMPOSANTS
Session 158 - page 54

© 2003-2018 CARAPACE



Gestion de projets informatique complexes

5.1

Partons des Use Cases

■ Allons plus loin

- Il n'y a pas de différence entre un diagnostic, une ordonnance, un acte
- Il s'agit de remplir un document pré-formaté, en remplissant :
 - Des champs obligatoires (dont certains sont automatiquement renseignés ; date, prescripteur, nom du patient...)
 - Des champs facultatives
 - Des champs libres
- En d'autres termes, il y a une seule commande : « Editer un document », dans lequel on passe un pointeur sur un template

5.1

Partons des Use Cases

■ Cela commence à se décanter

Composant CONNEXION	
Cmd1	Se connecter
Cmd2	Se déconnecter

Composant PROFIL	
Cmd1	Modifier un profil

Composant DOCUMENT	
Cmd1	Editer un document
	Diagnostic
	Ordonnance
	Compte Rendu Acte

Composant AGENDA	
Cmd1	Lire/Modifier un agenda

Composant DMP	
Cmd1	Ouvrir un DMP
Cmd2	Consulter les documents

Composant MEDECIN	
Cmd1	Demander un examen
Cmd2	Coder des actes

■ Ca devient sympa

5.1

Partons des Use Cases

- **On voit maintenant ce qui va nous manquer**
 - Des composants pour afficher les informations et les saisir
 - Des composants pour gérer les objets
 - Base des utilisateurs - Profil - Agenda
 - DMP - Documents...
 - Un composant qui sauvegarde les informations créées ou modifiées en mémoire dans la base et qui les remonte en mémoire depuis la base
 - Un composant chef d'orchestre qui reçoit les événements et qui met à jour la présentation et la base de données : le Contrôler
 - Des templates de documents
 - Des listes d'icônes – Des fichiers de messages
 - Des Listes de composants
 - Des règles de contrôle d'intégrité

MODELE DE COMPOSANTS

Session 158 - page 57

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



5.2

Construisons la présentation

- **Hypothèse**
 - Dans ce qui suit, je travaille sur une IHM pour ordinateur, comportant une souris et un clavier
 - Dans ce qui suit, je fais l'hypothèse que l'écran est partagé en 3 zones
 - Une barre horizontale en haut pour des commandes permanentes
 - Une barre verticale à gauche pour des commandes contextuelles
 - Une zone centrale, la plus étendue possible, d'affichage de l'information en cours de traitement ou de consultation



MODELE DE COMPOSANTS

Session 158 - page 58

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



5.2

Construisons la présentation

- **Une barre horizontale en haut pour des commandes permanentes**
 - Nom du personnel hospitalier connecté
 - Déconnexion
 - Gestion de profil
 - Nom du patient traité
 - Agenda
 - Help
 - Information sur l'AP-HP
 - Information sur l'entreprise éditrice du logiciel



5.2

Construisons la présentation

- **Chaque zone sélectionnable de cette barre horizontale c'est**
 - Une icône
 - Un texte explicatif qui apparait quand on laisse la souris sur l'icône une demi seconde (sur PC)
 - Un pointeur vers
 - Un composant
 - Une entry
 - Des paramètres contextuels



5.2

Construisons la présentation

- **Le composant qui gère cette barre horizontale a plusieurs points d'entrée**
 - Remise à zéro de la zone
 - Affichage du nom du personnel hospitalier connecté et de l'icône Déconnexion
 - Affichage du nom du patient traité
 - Affichage de l'icône Agenda
 - Affichage de l'icône Help
 - Affichage de l'icône de l'AP-HP
 - Affichage de l'icône de l'entreprise éditrice du logiciel



5.2

Construisons la présentation

- **Une barre verticale à gauche pour les commandes contextuelles.**
- **Par exemple pour un médecin**
 - Finaliser les documents en cours
 - Ouvrir un DMP
 - Demander un examen
 - Définir un Diagnostic
 - Rédiger une ordonnance
 - Rédiger un compte rendu d'acte (en version V2 ou V3)
 - Coder des actes (en version V2 ou V3)



5.2

Construisons la présentation

- **Certains articles de menu n'ont pas de sens**
 - C'est le cas de
 - « Demander un examen »
 - « Définir un diagnostic »
 - « Définir une posologie »
 - « Faire un compte rendu d'acte médical »
 - Tant que le patient n'a pas été choisi.
- **Vous avez 3 possibilités**
 - Soit faire apparaître un message d'erreur lorsque le médecin sélectionne une de ces icônes
 - Soit griser lesdites icônes pour les rendre in-sélectionnables
 - Soit les faire disparaître, et réapparaître lorsqu'un patient est choisi

5.2

Construisons la présentation

- **Chaque zone sélectionnable de cette barre verticale c'est**
 - Une icône
 - Un texte explicatif qui apparaît quand on laisse la souris sur l'icône une demi seconde (sur PC)
 - Un pointeur vers
 - Un composant
 - Une entry
 - Des paramètres contextuels

5.2

Construisons la présentation

- **Le composant qui gère cette barre verticale a deux points d'entrée**
 - Remise à zéro de la zone
 - Affichage d'une nouvelle icône qui vient se mettre à la suite de la dernière icône affichée
 - On envoie
 - Une icône
 - Un texte explicatif qui apparaît quand on laisse la souris sur l'icône une demi seconde (sur PC)
 - Un pointeur vers
 - Un composant
 - Une entry
 - Des paramètres contextuels
 - Remarque : plusieurs icônes peuvent pointer vers le même composant et la même entry



5.2

Construisons la présentation

- **Une zone centrale, la plus étendue possible, d'affichage de l'information en cours de traitement ou de consultation**
 - Le panel de connexion
 - Le profil du personnel hospitalier connecté ou du patient traité
 - Un agenda
 - La documentation User
 - La documentation de votre entreprise
 - L'arborescence du DMP
 - Le document en consultation
 - Le document en cours d'édition...




Construisons la présentation



Login :

Password :





Gestion de projets informatique complexes



Construisons la présentation



Gestion de projets informatique complexes



Construisons la présentation



5.2

Construisons la présentation

Assistance Médicale
HÔPITAL DE PARIS

Antoine Celier Ch. 304 Alois Alzheimer

Publier

Choisir un patient

A propos de de l'AP-HP

Documents non finalisés

Définir un diagnostic

Rédiger une ordonnance

Rédiger un acte

Demander un examen

Editer le profil du patient

Editer mon profil

Déconnexion

Planning

Help

A propos de notre entreprise

© 2003-2018 CARAPACE

Session 158 - page 71

Gestion de projets informatiques complexes

5.2

Construisons la présentation

Assistance Médicale
HÔPITAL DE PARIS

Antoine Celier Ch. 304 Alois Alzheimer

Publier

Date Service Responsable Typologie

2018-10-26	Cardiologie	P. Assayag	Consultation	
2018-10-26	Cardiologie	N.Cointat	Electro-Card	
2018-06-17	Cardiologie	P. Assayag	Electro-Card	
2018-06-17	Cardiologie	N.Cointat	Electro-Card	
2018-06-12	Cardiologie	D. Zeitouni	Test Effort	
2018-06-10	Cardiologie	F.Chleir	EchoDop TSA	
2018-06-10	Cardiologie	A.Pottier	Echog Card	
2017-11-09	Cardiologie	P. Assayag	Consultation	
2017-11-09	Cardiologie	N.Cointat	Electro-Card	
2017-04-19	Cardiologie	P. Assayag	Consultation	

© 2003-2018 CARAPACE

Session 158 - page

Gestion de projets informatiques complexes

5.2 Construisons la présentation

ASSISTANCE
PARCOURS
D'ÉVALUATION DE PARIS

Antoine Celier

Ch. 304

Alois Alzheimer

HELP!

CARAPACE

Publish

New Meeting

Appointment Meeting Items - New

Stages Meeting

Today

Day

Week

Month

Calendar

Calendar - Groups

Calendar - Groups - Manage Calendars

Find

Address Book

4 octobre 2018

LI

MA

ME

JE

VE

SA

DI

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

5 novembre 2018

LU

MA

ME

JE

VE

SA

DI

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

1

2

3

4

5

6

7

8

9

My Calendars

Calendar

New Group

Shared Calendars

RAMEL Pierre-Yves

HACHEMI Houman

GOETTY Philippe

BALESTRIER Anne

BARON LABAT Dominique

TAILLEUR Marie-Clémence

Other Calendars

STOESSER Philippe

WALLEY Laurent

MACHESLON

ACHES Ladoux (CONTRACTOR)

MICHAUD Florence

5 - 11 novembre 2018

Paris, Ville de Paris

Today 40° / 10°

Tomorrow 40° / 10°

Search Calendar (Ctrl-F)

5

6

7

8

9

10

11

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

LIQUID

5.2

Construisons la présentation

Antoine Celier

Ch. 304

Alois Alzheimer

Carapace

Publication

Aide

Aide

Aide

Aide

Laboratoire Maison Blanche

10 rue de la République - 92000 Nanterre
Tél : 01 41 39 12 34 - Fax : 01 41 39 12 35
E-mail : info@maison-blanche.fr

Laboratoire Unibio 92

10 rue de la République - 92000 Nanterre
Tél : 01 41 39 12 34 - Fax : 01 41 39 12 35
E-mail : info@unibio92.fr

Laboratoire Maison Blanche

Date : 2024-10-20

Don de sang : 29/10/2024

Date de naissance : 14/10/1984 à 17:59

Donneur : 123456789

Le 14/10/2028

BIOPTIE RECTALE

HEMATOLOGIE

REMARQUE

Donneur 123456789, Cytomètre en cours de maintenance (non)

Anémie (déficit ferritin < 100 ng/L) (2024-10-20) (pauvre en fer) (Système sanguin) (S2)

Paramètre	Unité	Resultat	Normale	Commentaire
Formules globales				
Leucocytes	6.56	6.56 (10.0-12.0)	4.00-10.00	2024-10-20
Neutrophiles	4.80	4.80 (4.0-10.0)	4.00-10.00	17.9
Monocytes	1.14	1.14 (0.0-1.0)	0.0-1.0	1.8
Eosinophiles	0.21	0.21 (0.0-1.0)	0.0-1.0	0.3
Plaquettes	27.8	27.8 (15.0-40.0)	15.0-40.0	84.8
Conc. hém. en Hb (SD-SC)	32.8	32.8 (30.0-35.0)	30.0-35.0	83.3
Formules récapitulées				
Pop. neutrophiles	42.2	42.2 %	40.0-60.0	14.7
Pop. monocytes	17.0	17.0 %	10.0-20.0	6.6
Pop. plaquettes	12.7	12.7 %	10.0-15.0	1.8
Pop. éosinophiles	2.1	2.1 %	0.0-5.0	0.3
Pop. lymphocytes	23.9	23.9 %	15.0-40.0	1.0
Plaquettes	10.5	10.5 %	0.0-1.0	0.4
Formules plaquettes				
Plaquettes	261	261 G/L	150-400	2024-10-20
Donneur 123456789	13.7	13.7	10.0-15.0	1.8

BIOCHIMIE SÉRIQUE

Cholestérol

Donneur 123456789

Donneur 123456789

Donneur 123456789

Donneur 123456789

10 rue de la République - 92000 Nanterre
Tél : 01 41 39 12 34 - Fax : 01 41 39 12 35
E-mail : info@maison-blanche.fr

Donneur 123456789

10 rue de la République - 92000 Nanterre
Tél : 01 41 39 12 34 - Fax : 01 41 39 12 35
E-mail : info@unibio92.fr

Laboratoire Maison Blanche

Date : 2024-10-20

Don de sang : 29/10/2024

Date de naissance : 14/10/1984 à 17:59

Donneur : 123456789

Le 14/10/2028

BIOPTIE RECTALE

HEMATOLOGIE

REMARQUE

Donneur 123456789, Cytomètre en cours de maintenance (non)

Anémie (déficit ferritin < 100 ng/L) (2024-10-20) (pauvre en fer) (Système sanguin) (S2)

Paramètre	Unité	Resultat	Normale	Commentaire
Formules globales				
Leucocytes	6.56	6.56 (10.0-12.0)	4.00-10.00	2024-10-20
Neutrophiles	4.80	4.80 (4.0-10.0)	4.00-10.00	17.9
Monocytes	1.14	1.14 (0.0-1.0)	0.0-1.0	1.8
Eosinophiles	0.21	0.21 (0.0-1.0)	0.0-1.0	0.3
Plaquettes	27.8	27.8 (15.0-40.0)	15.0-40.0	84.8
Conc. hém. en Hb (SD-SC)	32.8	32.8 (30.0-35.0)	30.0-35.0	83.3
Formules récapitulées				
Pop. neutrophiles	42.2	42.2 %	40.0-60.0	14.7
Pop. monocytes	17.0	17.0 %	10.0-20.0	6.6
Pop. plaquettes	12.7	12.7 %	10.0-15.0	1.8

5.2

Construisons la présentation

Antoine Celier Ch. 304 Alois Alzheimer



Abandon Sauvegarder comme brouillon Publier

MODELE DE COMPOSANTS © 2003-2018 CARAPACE
Session 158 - page 75 Gestion de projets informatique complexes

5.3

Gérons les objets

- **Les composants objets ont pour objectif**
 - De gérer la structure des objets en mémoire
 - De communiquer avec le composant « Dépôt de Données » pour
 - Charger des objets en mémoire
 - Sauvegarder des objets en base
 - Transmettre une description standard de l'objet à un composant Business Unit ou Présentation
 - Recevoir des demandes de modification de l'objet d'un composant Business Unit ou Présentation

- **Un composant objet**

- Communique avec le composant Dépôt de Données
- Au travers d'une description standard de l'objet qui peut être
 - Eloignée de la structure de l'objet en mémoire
 - Eloignée de la structure de l'objet en Base

- **Un composant objet**

- Communique avec un composant Business Unit ou Présentation
- Au travers d'une description standard de l'objet qui peut être éloignée de la structure de l'objet en mémoire

- **Le composant DMP transmet une description standard de l'objet à un composant Business Unit ou Présentation**

- Il s'agit d'une liste d'enregistrements structurés comme suit :
 - Une date
 - Un identifiant de service
 - Le nom d'un responsable
 - Une typologie d'acte
 - L'identifiant d'un document structuré
 - Potentiellement l'identifiant d'un ou de deux documents complexes de type Imagerie
- Cette liste est classée en fonction d'un paramètre d'entrée (date, responsable, typologie d'acte ...)

5.3

Gérons les objets

- **Le composant DMP transmet une description standard de l'objet au composant Dépôt de Données**
- **Il s'agit**
 - D'un pointeur sur un patient
 - D'une liste de nouveaux enregistrements, ou d'enregistrements complétés, structurés comme suit :
 - Une date
 - Un pointeur vers un objet de l'arborescence de l'Hôpital
 - Un pointeur vers un membre du personnel hospitalier
 - Une pointeur vers une typologie d'acte
 - Un pointeur vers un document structuré
 - Potentiellement un ou deux pointeurs vers un document complexe de type Imagerie
 - Chaque enregistrement est marqué « Published = YES/NO »



5.3

Gérons les objets

- **Le composant DMP reçoit la description standard du DMP d'un patient depuis le composant Dépôt de Données**
- **Il s'agit**
 - D'une liste d'enregistrements structurés comme suit :
 - Une date
 - Un pointeur vers un objet de l'arborescence de l'Hôpital
 - Un pointeur vers un membre du personnel hospitalier
 - Une pointeur vers une typologie d'acte
 - Un pointeur vers un document structuré
 - Potentiellement un ou deux pointeurs vers un document complexe de type Imagerie
 - Chaque enregistrement est marqué « Published = YES/NO »



5.4

Examinons notre Archi. Logique

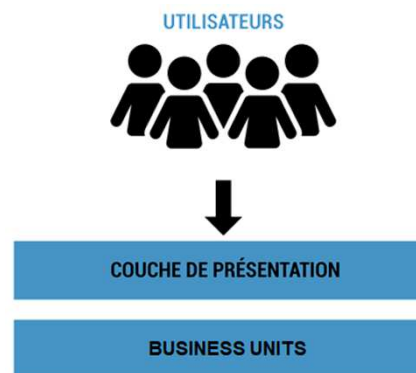
- Les composants de présentation vont permettre de
 - Afficher des informations
 - Créer, Modifier ou Supprimer une partie de cette information
 - Construire une requête



5.4

Examinons notre Archi. Logique

- Les composants de Business Unit vont permettre de suivre un workflow dont le résultat est généralement d'obtenir une information complète et cohérente qui pourra être publiée en Base de Données

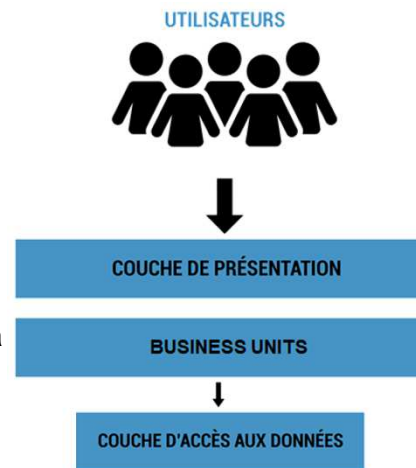


5.4

Examinons notre Archi. Logique

- Les composants Objet vont permettre de créer, modéliser et updater un objet en mémoire.

- Cet objet peut être créé ex nihilo au travers de l'application
- Cet objet peut être créé à partir d'un template
- Ou être chargé depuis la base de données

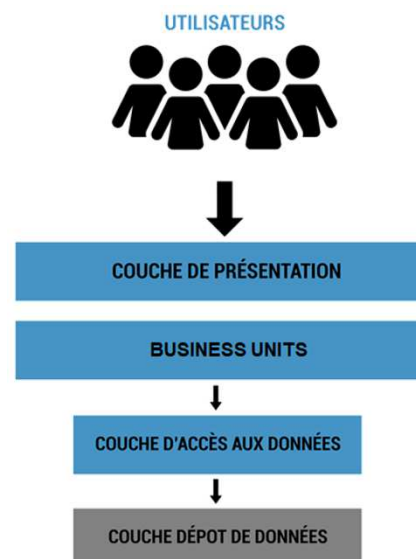


5.4

Examinons notre Archi. Logique

- Le composant de Data Management va permettre de :

- Identifier les objets de la Base de Données solution d'une requête
- Charger depuis la base de données des objets en mémoire, c'est-à-dire vers des composants Objet
- Sauvegarder en Base de Données des objets gérés en mémoire par des composants Objet

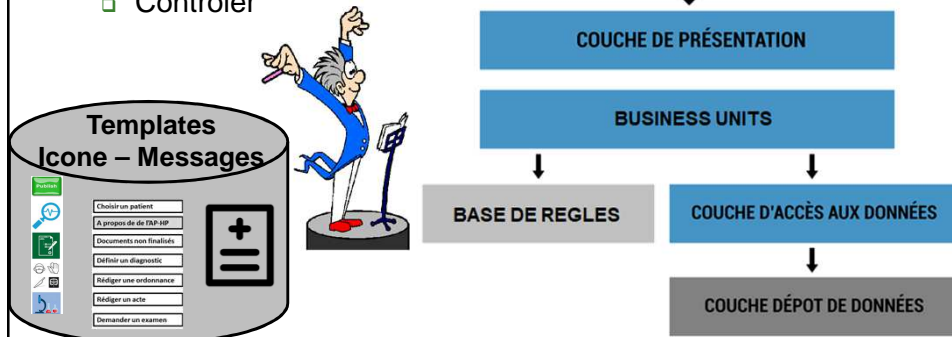


5.4

Examinons notre Archi. Logique

■ La réalité est un peu plus complexe :

- ❑ Templates
- ❑ Icônes et messages
- ❑ Base de Règles
- ❑ Contrôler



MODELE DE COMPOSANTS

Session 158 - page 85

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



6

Facteurs influençant l'A. L.

- Nous traiterons de la gestion des objets dans une présentation ultérieure
 - ❑ Continuons néanmoins à analyser notre solution
- Nous allons voir apparaître quelques points à prendre en compte pour atteindre les objectifs non fonctionnels du projet
 - ❑ Respect du secret médical
 - ❑ Cryptage des données
 - ❑ NLS
 - ❑ Help
 - ❑ Gestion des erreurs
 - ❑ Fin d'une commande

MODELE DE COMPOSANTS

Session 158 - page 86

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



6.1

Respect du secret médical

- **Dans le projet DMP, le respect du secret médical est une contrainte majeure.**
 - Cette contrainte forte ne se résout pas d'une façon unique et localisée
 - Mais irradie toute votre architecture
- **La connexion**
 - Seuls les users connus sont acceptés
- **L'accès au DMP**
 - Seuls les users ayant un attribut « Secret médical = YES » peuvent ouvrir un DMP



6.1

Respect du secret médical

- **De la même façon, la génération des documents est liée au rôle de chacun.**
 - Seul un médecin ou un interne peut définir un diagnostic ou une posologie
 - Seul un médecin ou un interne ayant la spécialité de cardiologie pourra définir un diagnostic de cardiologie.



6.2

Le cryptage des données

- **Le cryptage des données**
 - Toutes les données stockées en base doivent être cryptées
 - Toutes les données qui transitent sur le Net, doivent être cryptées
 - Cela va vous conduire à identifier assez rapidement où s'exécutent chacun de vos composants
- **Il existe d'autre mécanisme de sécurité :**
 - Résistance face au hacking et à l'injection de code : Malware, Trojan, Worm, Spyware
 - Nous verrons cet aspect dans la partie "Architecture Physique"



6.3

NLS

- **Un nombre important de membres du personnel hospitalier de l'AP-HP est d'origine étrangère**
 - Sous- effectif chronique
 - Salaire en deçà de celui pratiqué en clinique privée
 - Conditions de travail assez éreintantes
- **Dans ce cadre, il peut être pertinent de proposer le logiciel dans la langue du membre hospitalier**
- **La langue de dialogue se manifeste essentiellement**
 - Par les champs dans les panels de dialogue
 - Par les champs dans les menus ou les commandes permanentes
 - Par les textes apparaissant lorsqu'on passe la souris sur une icône



- **Si**
 - Vos champs de panels, de menus, de commandes permanentes ou de textes associés aux icones sont en dur dans le code, cela va être difficile de changer de langue
 - Vos champs de panels, de menus, de commandes permanentes ou de textes associés aux icones sont en dur dans le code
 - Cela va être cher de les externaliser
 - Et cela va vous entrainer des régressions le jour où vous allez le faire

- **La solution consiste**
 - A référencer chaque message par un identifiant
 - Et à aller chercher le libellé du texte du message dans un fichier externe
- **Ceci permettra de changer le fichier de message en fonction de la langue de l'utilisateur**
- **Et cette action est à mettre en place dès la version 1 de votre logiciel**
 - Ce qui ne signifie pas que vous allez livrer les fichiers de message de toutes les langues.
 - Vous n'en avez ni le savoir ni le temps
 - Vous offrirez les outils pour générer le fichier d'une langue étrangère à partir du fichier de langue française

■ Attention :

- ❑ Le fait de changer de langue conduit à avoir des champs à géométrie variable
- ❑ En règle général le NLS (National Language Support) n'accepte pas les langue écrivant de droite à gauche
 - Arabe
 - Divehi (Maldivien)
 - Hébreu
 - Ourdou (langue officiel du Paistan)
 - Persan (Araméen)
 - Syriaque
 - Yiddish

■ Mettre à disposition une documentation utilisateur papier ne sert généralement pas à grand-chose

- ❑ Les utilisateurs ne savent plus où elle est le jour où ils en ont besoin
- ❑ Du fait de correctifs et d'évolutions, elle n'est généralement jamais à jour
- ❑ Difficile de l'avoir à coté de soi si on utilise le produit sur Smartphone ou sur tablette
- ❑ Et imprimer une doc de 20 pages à 100 000 exemplaire représente la bagatelle d'une pile de feuilles A4 de 200 mètres de haut.

- **A minima, la doc est une doc en ligne**
 - Elle est stockée sur le serveur de données
 - Et elle est appelée à tout moment comme une commande transparente.
- **Mais votre Architecture Fonctionnelle définit une arborescence de commande.**
 - Il est à penser que votre documentation user va être construite autour de cette hiérarchie de commandes.
 - Il peut donc être pertinent de positionner la documentation sur la page de la commande en cours lorsque vous sollicitez la commande Help

- **Dans le modèle CUPRIMD, la convivialité (Usability) est également une qualité majeure**
- **Dans ce cadre, le travail du Designer est important**
 - Respect de Standards
 - Faible nombre d'interactions
 - Les mêmes faits produisent les mêmes effets
 - Mieux vaut un choix dans une liste que la saisie d'une chaîne de caractères....
- **Mais cela n'empêche pas l'utilisateur de faire des erreurs**

6.5

Gestion des erreurs

- **Faire une erreur peut être déstabilisant pour certains**
 - Traumatisme de jeunesse
 - Si, qui plus est, la façon de comprendre puis de contourner l'erreur n'a rien d'évident
 - Cela peut conduire à une frustration importante qui conduit à un rejet du produit
- **Pas question pour autant de laisser l'utilisateur faire des erreurs**
 - Le respect du secret médical, et l'intégrité des données, sont des priorités absolues

6.5

Gestion des erreurs

- **Ce qu'il faut c'est**
 - Prévenir intelligemment l'utilisateur de la typologie de l'erreur
 - Et lui offrir un guide pour contourner l'erreur, qu'il pourra par exemple faire apparaître en sélectionnant une icône à la fin du message d'erreur.

Erreur ~~2817~~

Erreur : le numéro de Sécurité Sociale est invalide →

Le numéro de Sécurité Sociale

- Doit commencer par un 1 ou un 2
- Les 2 chiffres suivants correspondent à l'année de naissance du patient
- Les deux chiffres suivants correspondent au mois de naissance du patient
- Les deux chiffres suivants correspondent au département de naissance du patient
- Il comporte 13 chiffres (plus 2 chiffres)

6.5

Gestion des erreurs

- Dans une version très évoluée de la gestion d'erreur, les paramètres en cours de l'application peuvent être utilisés pour mettre en évidence l'erreur

Numéro : 104	Rue : Lasègue
CP : 92320	Ville : Montrouge

Erreur : adresse invalide →

92320 n'est pas le code postal de **Montrouge**
Il n'y a pas de rue **Lasègue** à **Montrouge**

6.6

Fin d'une commande

- Il y a deux type de commandes
 - Les commandes workflow
 - Les commandes transparentes
- **Commande Workflow** : elle déroule un (mini-processus)
- **Commande transparente** : elle permet de consulter et de modifier une information tout en conservant le contexte de l'application

6.6

Fin d'une commande

- **Exemple de commande Workflow : menu du médecin « Définir un Diagnostic ».**
 - Une liste de types de diagnostics apparaît
 - Le médecin sélectionne un type de diagnostic
 - Un panel standard apparaît, avec :
 - Un certain nombre de champs prévalués
 - Un certain nombre de champs obligatoires
 - Un certain nombre de champs optionnels
 - Une zone de texte libre
 - Une icône « Abandon du diagnostic »
 - Le médecin commence à remplir les champs
 - Une seconde icône « Sauvegarder comme brouillon » apparaît
 - Une fois tous les champs obligatoires renseignés, une troisième icône apparaît : « Publier en base.

MODELE DE COMPOSANTS

Session 158 - page 101

© 2003-2018 CARAPACE

Gestion de projets informatique complexes



6.6

Fin d'une commande

- **« Définir un Diagnostic » (2)**
 - Dès lors que le médecin clique sur une des icônes
 - « Sauvegarder en brouillon »
 - « Publier en Base »
 - « Abandon du Diagnostic »
 - Le dialogue se repositionne au début de la commande
 - Une liste de types de diagnostics apparaît
- **Il y a donc une notion de document en édition en cours**
 - Qui est matérialisé dès que le médecin commence à saisir de l'information
 - Le fait de changer de commande workflow déclenche de facto la sauvegarde du document comme brouillon
 - Il en est de même lors de la déconnexion

MODELE DE COMPOSANTS

Session 158 - page 102

© 2003-2018 CARAPACE

Gestion de projets informatique complexes



6.6

Fin d'une commande

- **Exemples de commande transparente:**
 - Consulter son agenda (Icône « Close »)
 - Editer son profil (Icones « Cancel » et « Close »)
 - Sélectionner l'aide en ligne (Icône « Close »)
- **Lorsque je termine une commande transparente**
 - Par sélection d'une icône « Cancel » ou « Close »
 - Je sauvegarde si besoin le profil en cours d'édition
 - Je reviens à l'endroit exact du dialogue ou je me trouvais avant la commande transparente.

MODELE DE COMPOSANTS

Session 158 - page 103

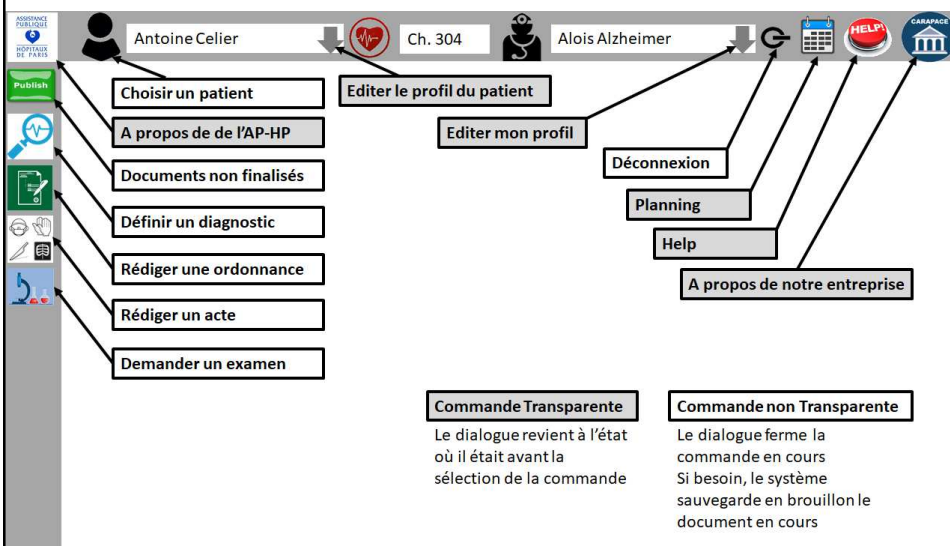
© 2003-2018 CARAPACE

Gestion de projets informatique complexes



6.6

Fin d'une commande



MODELE DE COMPOSANTS

Session 158 - page 104

© 2003-2018 CARAPACE

Gestion de projets informatique complexes



7

Où en sommes-nous ?

- Au travers de ce premier voyage que je vous ai fait faire
 - J'ai utilisé plusieurs principes du Génie Logiciel
 - Architecture en couches
 - Architecture Orientée objet
 - J'ai fait émerger un composant de présentation relativement générique réutilisable
 - J'ai transformé des composants procéduraux en composants déclaratifs
 - J'ai fusionné un certain nombre de composants
 - J'ai utilisé au maximum l'encapsulation

MODELE DE COMPOSANTS
Session 158 - page 105

© 2003-2018 CARAPACE

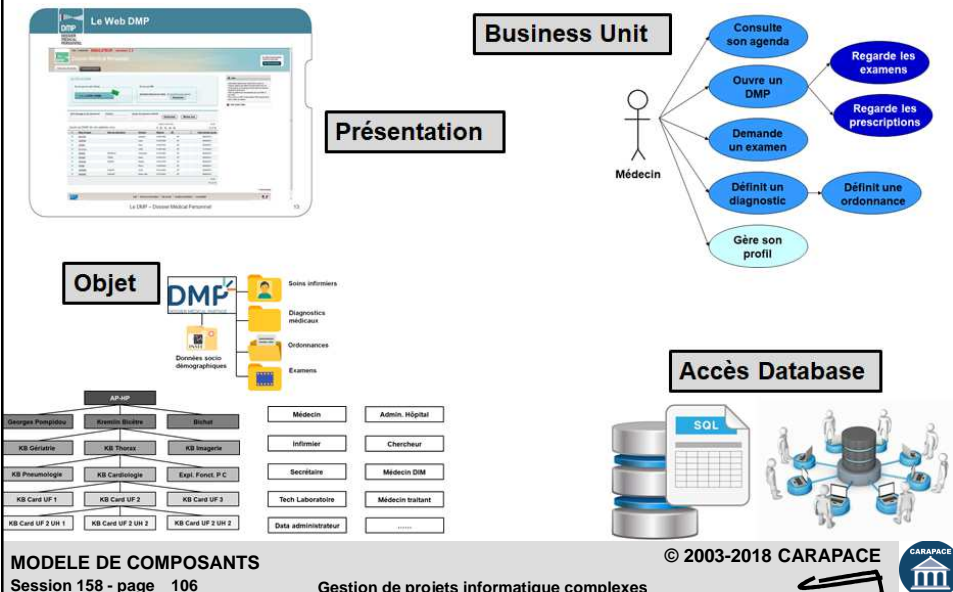


Gestion de projets informatique complexes

7

Où en sommes-nous ?

- J'ai bien séparé les composants en fonction de savoir-faire



MODELE DE COMPOSANTS
Session 158 - page 106

© 2003-2018 CARAPACE



Gestion de projets informatique complexes

7

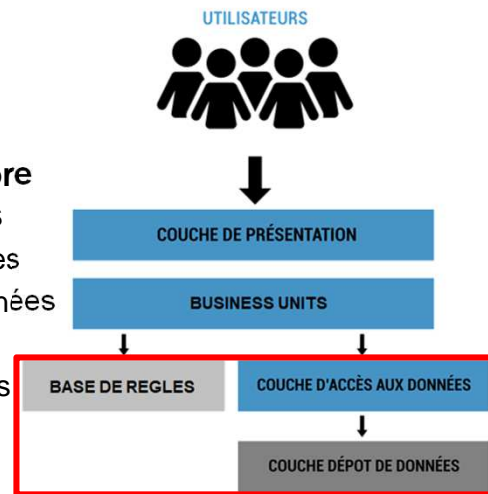
Où en sommes-nous ?

■ Bref, j'ai appliqué les méthodes de base du Génie Logiciel

- Approche Fractale
- Réutilisation
- Encapsulation

■ Je n'ai certes pas encore traité la partie données

- Couche Dépôt de données
- Couche d'accès aux données
- Base de Règles
- Que nous allons voir dans la présentation suivante



MODELE DE COMPOSANTS

Session 158 - page 107

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



7

Où en sommes-nous ?

■ Mais en l'état actuel, j'ai le sentiment que

- Mon produit pourrait s'enrichir dans les versions V2 et V3
- Sans remettre en cause fondamentalement mon Architecture de Composants

■ Ce n'est certes qu'une première étape, qui demanderait à être enrichie.

« Avant donc que d'écrire, apprenez à penser.
Ce que l'on conçoit bien s'énonce clairement,
Et les mots pour le dire arrivent aisément.
Hâtez-vous lentement, et sans perdre courage,
Vingt fois sur le métier remettez votre ouvrage,
Polissez-le sans cesse, et le repolissez,
Ajoutez quelquefois, et souvent effacez. »



Nicolas Boileau in « L'Art poétique »

MODELE DE COMPOSANTS

Session 158 - page 108

Gestion de projets informatique complexes

© 2003-2018 CARAPACE



7

Où en sommes-nous ?

■ Le développement par l'Organisation

- ❑ Votre Organisation a du savoir faire (la maîtrise de techniques)
- ❑ Votre Organisation a un budget (un nombre d'heures)
- ❑ Chaque version doit avoir du sens (un ROI pour le client)

■ Vous avez

- ❑ Identifié tous les composants à développer
- ❑ Attribuer les composants (en fonction des savoir-faire)
- ❑ Estimé le coût de chacun d'eux

■ Il reste à en définir la planification avec 2 objectifs

- ❑ Obtenir, avec le responsable Planning et Budget, une charge équilibrée pour chaque membre de l'Organisation
- ❑ Prendre en compte la Gestion des Risques

MODELE DE COMPOSANTS

Session 158 - page 109

© 2003-2018 CARAPACE

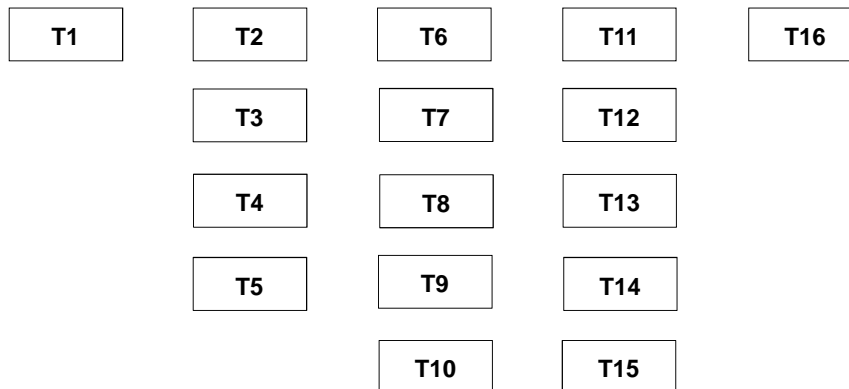
Gestion de projets informatique complexes



7

Où en sommes-nous ?

■ Vous avez identifié les composants



■ Vous avez identifié les savoir-faire et les technologies

MODELE DE COMPOSANTS

Session 158 - page 110

© 2003-2018 CARAPACE

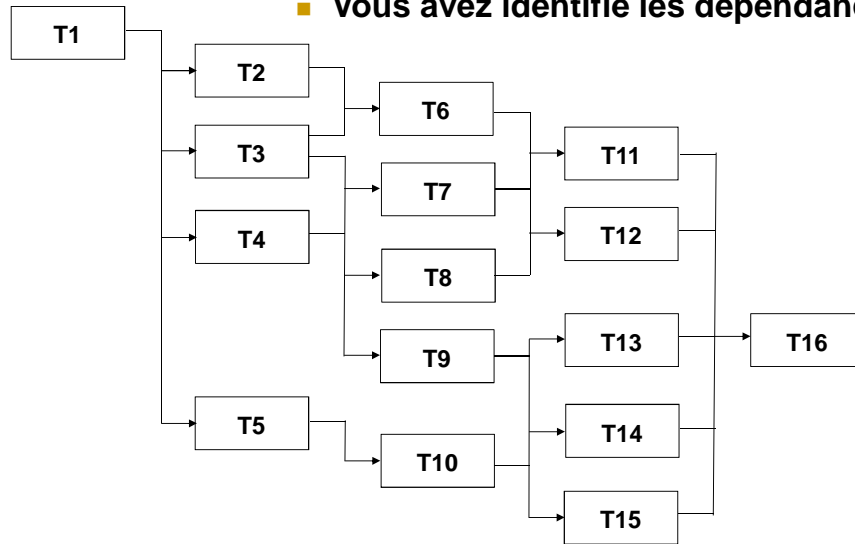
Gestion de projets informatique complexes



7

Où en sommes-nous ?

- Vous avez identifié les dépendances



MODELE DE COMPOSANTS

Session 158 - page 111

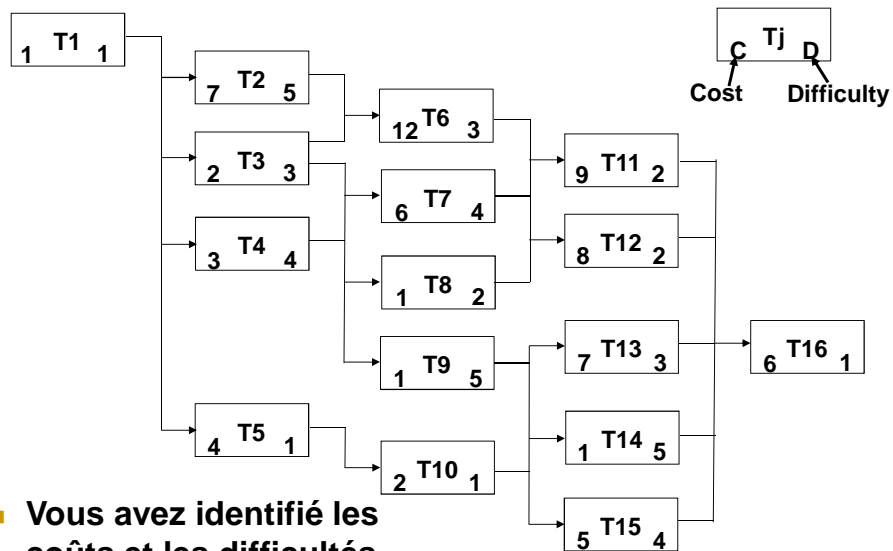
© 2003-2018 CARAPACE

Gestion de projets informatiques complexes



7

Où en sommes-nous ?



- Vous avez identifié les coûts et les difficultés

MODELE DE COMPOSANTS

Session 158 - page 112

© 2003-2018 CARAPACE

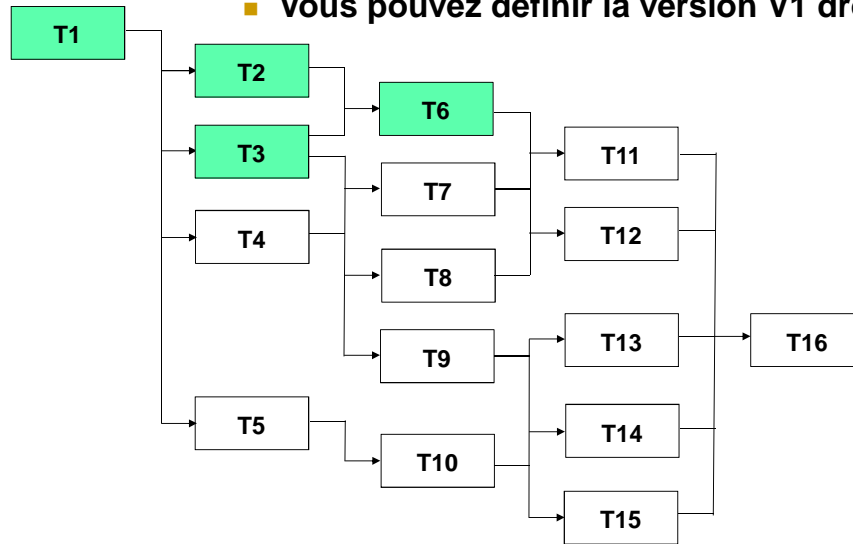
Gestion de projets informatiques complexes



7

Où en sommes-nous ?

- Vous pouvez définir la version V1 drop 1



MODELE DE COMPOSANTS
Session 158 - page 113

© 2003-2018 CARAPACE

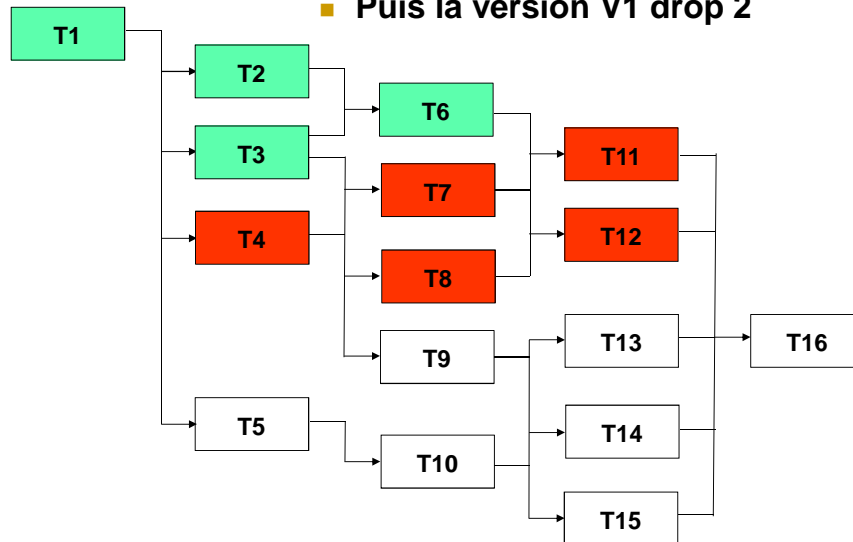
Gestion de projets informatique complexes



7

Où en sommes-nous ?

- Puis la version V1 drop 2



MODELE DE COMPOSANTS
Session 158 - page 114

© 2003-2018 CARAPACE

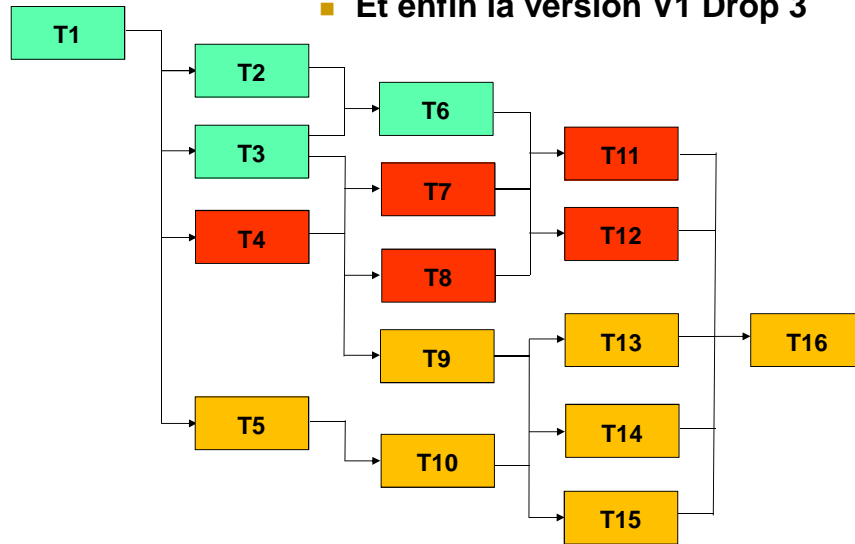
Gestion de projets informatique complexes



7

Où en sommes-nous ?

■ Et enfin la version V1 Drop 3



MODELE DE COMPOSANTS

Session 158 - page 115

Gestion de projets informatique complexes

© 2003-2018 CARAPACE

