


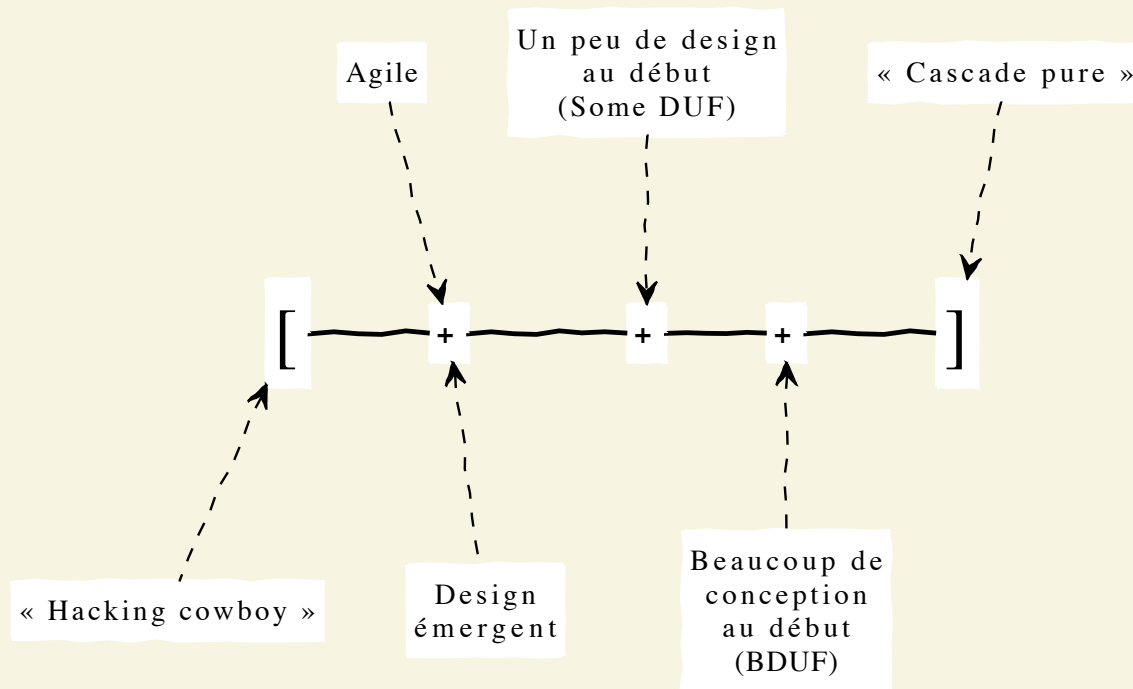
# LOG210 SÉANCE #11

## OBJECTIF DE LA SÉANCE

1. Dette technique  51.06m
2. Rappel architecture en couches
3. Diagrammes d'état
4. Diagrammes d'activité
5. Processus Unifié (développement itératif et incrémental)
6. GRASP dans les GoF

# DETTE TECHNIQUE

Rappel : Spectre de la conception de Neal Ford.



# DETTE TECHNIQUE : RISQUE

La dette est une forme de risque.

Elle apporte des *bénéfices* ou des *pertes*, selon la quantité d'intérêt à payer.

Chapitre 18 des notes de cours

# DETTE TECHNIQUE: MÉTAPHORE

Le “Hacking Cowboy” 🤠 (s'endetter)

- Bénéfice: obtenir une solution rapidement
- Perte: utiliser ce code à long terme

Pour *rembourser* la dette, on doit réécrire le code (faire un meilleur design, écrire des tests, “refactor”, etc.)

# DETTE TECHNIQUE: EXEMPLE

Accumulation de décision dans le temps qui donne un code de mauvaise qualité et contraint la solution architecturale

- Conséquences
  - Taux élevé de défaut
  - Retard du projet
  - Complication d'entretien
  - La productivité de développement très faible
- Proxy pour dette techniques
  - Combien de temps faut t'il pour mettre à jour le système
  - Combien coute la mise à jour du système.
  - Combien de fois les choses sont cassées.

# CLASSIFICATION DE LA DETTE TECHNIQUE

Fowler, 2009

Dette	Imprudente	Prudente
Délibérée	“On n’a pas le temps pour la conception”	“Faut livrer maintenant puis en assumer les conséquences”
Involontaire	“C’est quoi la séparation en couches?”	“Maintenant on sait comment on aurait dû le faire”

# DETTES TECHNIQUE: AUTRE EXEMPLES

- Ne pas migrer à la dernière version du framework de développement
- Ne pas faire d'analyse
- Ne pas faire de conception
- Ne pas faire de test
- Ne pas avoir de tracabilité entre les différents documents

# 5 ARGUMENTS POUR QUE LES MANAGERS SE SOUCIENT DE LA DETTE TECHNIQUE

1. Le réusinage réduira la volatilité du coût marginal des caractéristiques»
2. Au cours des 3 derniers mois, nous avons dépensé 63% du budget de développement pour régler des questions de qualité
3. Nous avons pris des prêts techniques pour expédier plus rapidement, nous devons rembourser une dette pour garder un délai de marché faible"
4. Nous pouvons réduire nos dépense d'affaires en investissant 10% de notre temps dans la qualité de notre code"
5. Investir 20% du budget dans le réusinage coupera le temps de première réponse de moitié avec un ROI positif sur la productivité de développeurs

réf.: <https://understandlegacycode.com/blog/5-arguments-to-make-managers-care-about-technical-debt/>

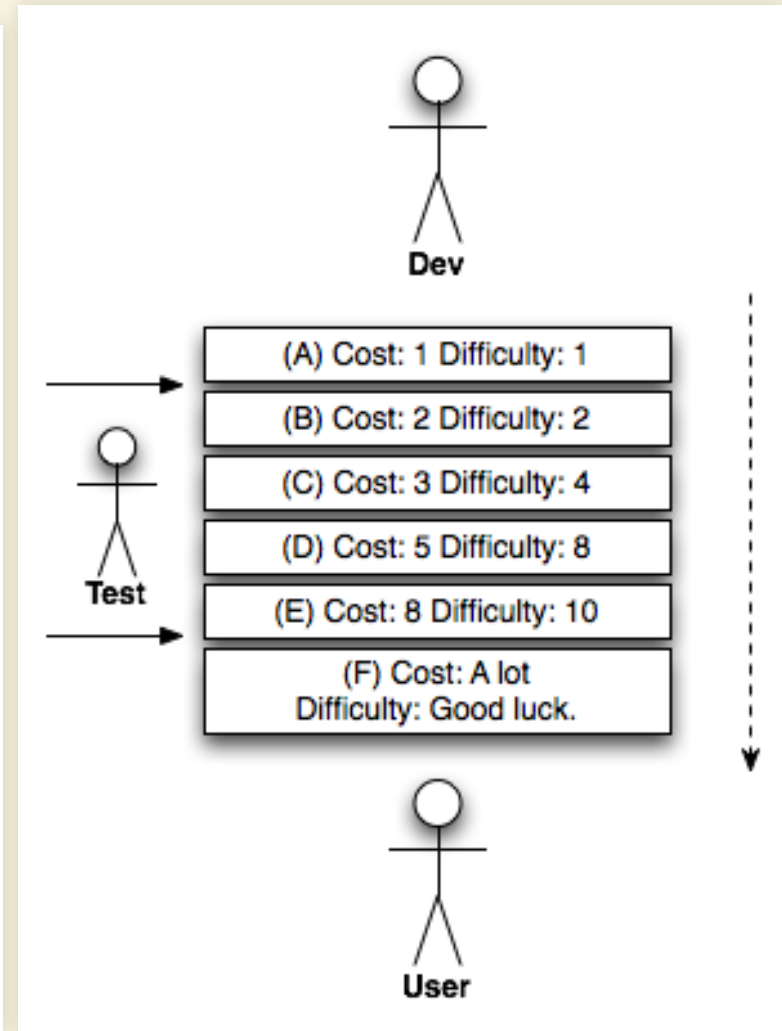
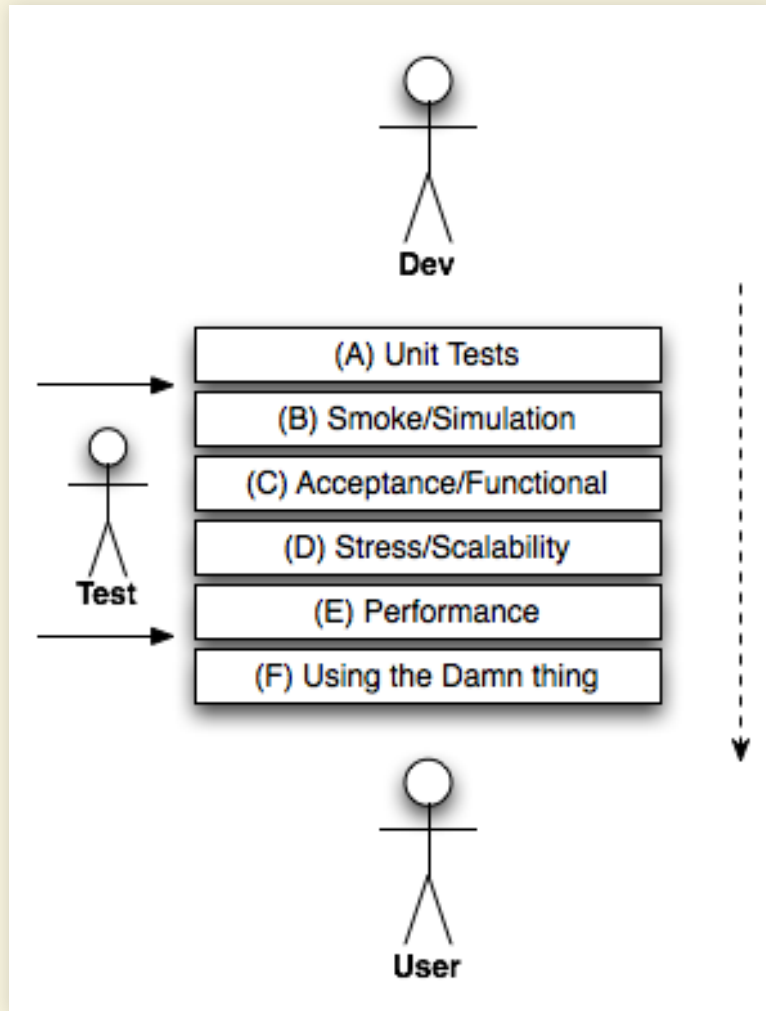


# DETTES TECHNIQUE: TESTS

- Essentiellement, il est dans votre intérêt, en tant que développeur, en tant qu'équipe, d'encourager de nombreux tests plus bas dans les piles présentées ici. Cela commence par des tests unitaires complets et vérifiés. Cela continue avec une discipline de test solide et reproductible (pour laquelle je recommande l'automatisation des tests).
- Pourquoi? Parce que - lorsque vous vous déplacez plus haut dans la pile, ce fichu bug que quelqu'un a enregistré est caché derrière couche après couche de code. Plus le bogue est éloigné du niveau d'unité, plus les composants et les variables d'environnement sont impliqués. Plus ceux-ci sont impliqués, plus il est difficile d'identifier et de réparer, et plus le coût est élevé.

<http://jessenoller.com/blog/2008/09/17/the-cost-of-not-testing-software>

# DETTES TECHNIQUE: TESTS

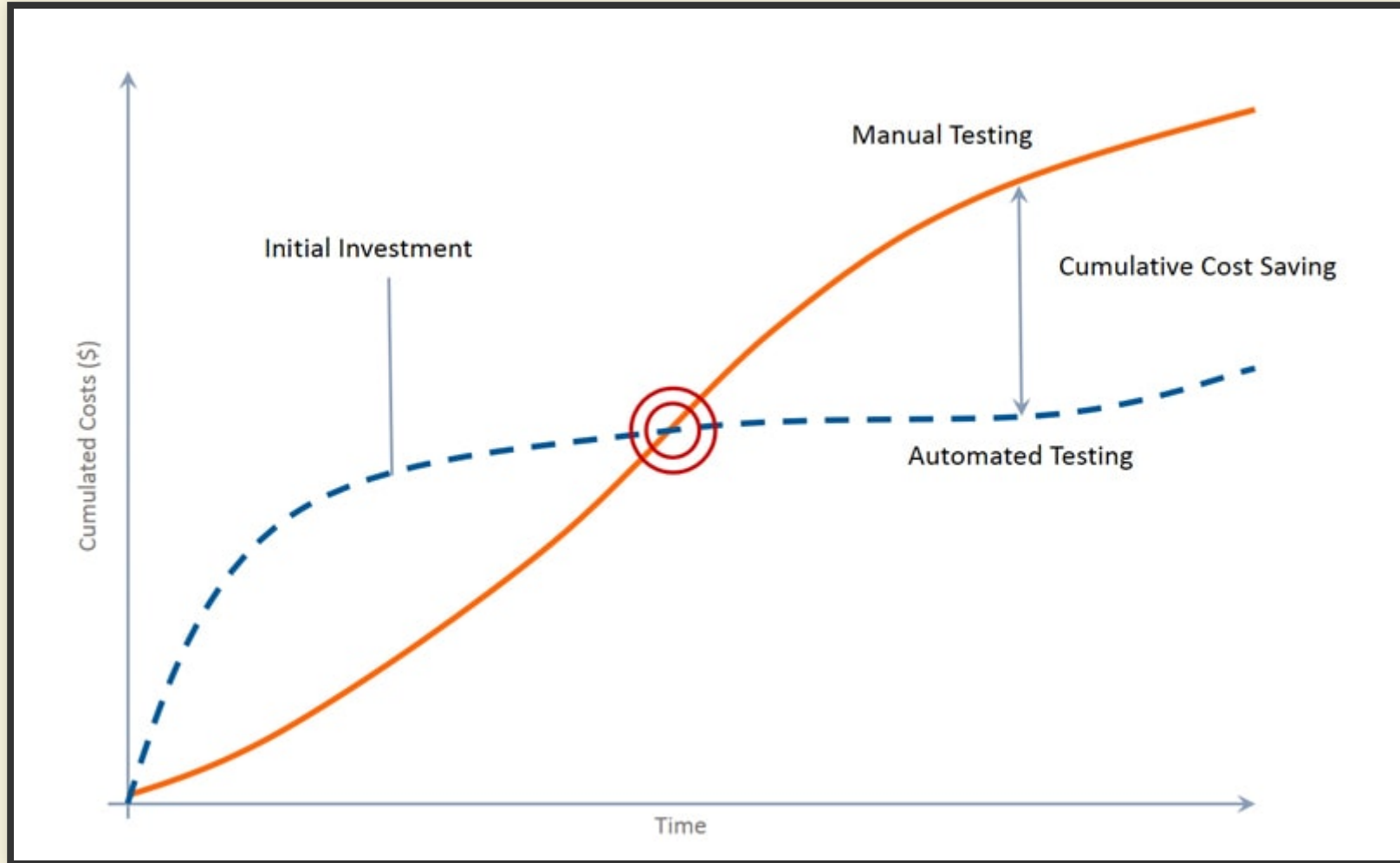


# DETTES TECHNIQUE: TESTS

## A 75\$/HEURE

	Coding/Unit Testing	Integration	Beta Testing	Post-Release
Hours to Fix	3.2	9.7	12.2	14.8
Cost to fix (\$)	240	728	915	1,110

# TEST MANUEL VS TEST AUTOMATISÉ



<https://abstracta.us/blog/test-automation/true-roi-test-automation/>

# AVANTAGES

- Un testeur manuel exécute des tests 8 heures par jour et rentre chez lui. À ce moment, les tests s'arrêtent. Avec l'automatisation des tests, nous pouvons exécuter des tests 16 heures de plus par jour (dans le meilleur des cas... bien sûr) pour le même coût, ce qui réduit le coût moyen des heures de test.

Manual	Automated
<b>Hours</b> $(10 \times 135) = 1,350 \text{ hours}$	<b>Hours</b> $(3 \times 21 \times 16 \times 5) + (7 \times 135) = \text{Total of } 5985 \text{ hours}$
<b>Cost</b> \$78/hour	<b>Cost</b> \$17.5/hour

- Coût total : 101250\$/mois
- 3 développeurs x 21 jours/mois \* 16h/jour de test \* 5 fois plus rapide = 5040h équivalent
- 7 développeurs x 135h/mois = 945h

# AVANTAGES

- Dans ce scénario, nous avons considérablement réduit le coût de chaque heure de test de 78 à 17,54, ce qui est un avantage que le directeur financier comprendra clairement. Ou vous pouvez le voir de cette façon; nous avons augmenté les tests de 1 350 heures à 5 985 heures équivalentes et gagné 315 000 \$ de tests par mois pour le même coût (5 040 fois le coût horaire moyen d'un testeur).

# VALEUR COMMERCIALE

- Améliorez la qualité des logiciels
- Évitez les problèmes de fonctionnement
- Maintenir une bonne image client
- Évitez les problèmes juridiques
- Diminuer le coût de la correction des bogues de 5 fois

# VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel



# VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel

Êtes-vous d'accord pour dire que le retour sur investissement de l'automatisation des tests est important?

# VISUAL STUDIO CODE

- voir plugin Tech Debt Metrics

# LOG210 SÉANCE #11

Copyright by Christopher Hopmann  
All Rights Reserved

## OBJECTIF DE LA SÉANCE

1. Dette technique
2. Rappel architecture en couches ← 10.19m
3. Dette technique vscode 4.09m
4. Diagrammes d'état
5. Diagrammes d'activité
6. Processus Unifié (développement itératif et incrémental)
- ≡ 7. GRASP dans les GoF

# ARCHITECTURE LOGIQUE

Valider l'architecture en couches (des laboratoires)

# LOG210 SÉANCE #11

Created by Rudez Studio  
from the Noun Project

## OBJECTIF DE LA SÉANCE

1. Dette technique
2. Rappel architecture en couches
3. Diagrammes d'état ← 2.54m
4. Diagrammes d'activité
5. Processus Unifié (développement itératif et incrémental)
6. GRASP dans les GoF

# DIAGRAMME D'ÉTAT



Created by Rudez Studio  
from the Noun Project

## Comment implémenter?

1. StackOverflow: Is there a typical state machine implementation pattern?
2. Derek Banas, State pattern

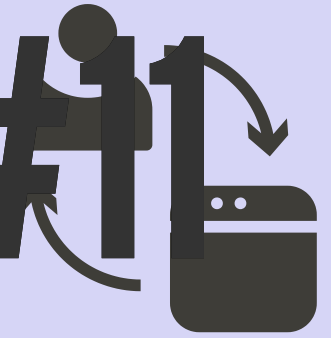
# DIAGRAMME D'ÉTAT

## EXERCICE VIDÉO PROJECTEUR

- En équipe de 4, réaliser le diagramme d'état en suivant les étapes identifiées dans l'exercice suivant:

<https://github.com/yvanross/LOG210-exercices/blob/master/etat/etat-videoprojecteur.md>

# LOG210 SÉANCE #11



## OBJECTIF DE LA SÉANCE

1. Dette technique
2. Rappel architecture en couches
3. Diagrammes d'état
4. Diagrammes d'activité ← 1.18m
5. Processus Unifié (développement itératif et incrémental)
6. GRASP dans les GoF



# DIAGRAMME D'ACTIVITÉ

A stylized icon of a person sitting at a desk with a computer, with an arrow pointing from the person to the computer, representing an activity diagram.

Comment implémenter?

Business Process Modeling Notation (BPMN)

1. BPMN Demo
2. UML
3. Derek Banas, Activity Diagram

# DIAGRAMME D'ACTIVITÉ

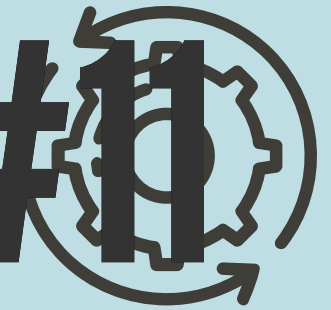


## EXERCICE RECETTE DE CUISINE

- En équipe de 4, réaliser le diagramme d'activité en suivant les étapes identifiées dans l'exercice suivant:

<https://github.com/yvanross/LOG210-exercices/blob/master/activite/activite-recette-de-cuisine.md>


# LOG210 SÉANCE #11



## OBJECTIF DE LA SÉANCE

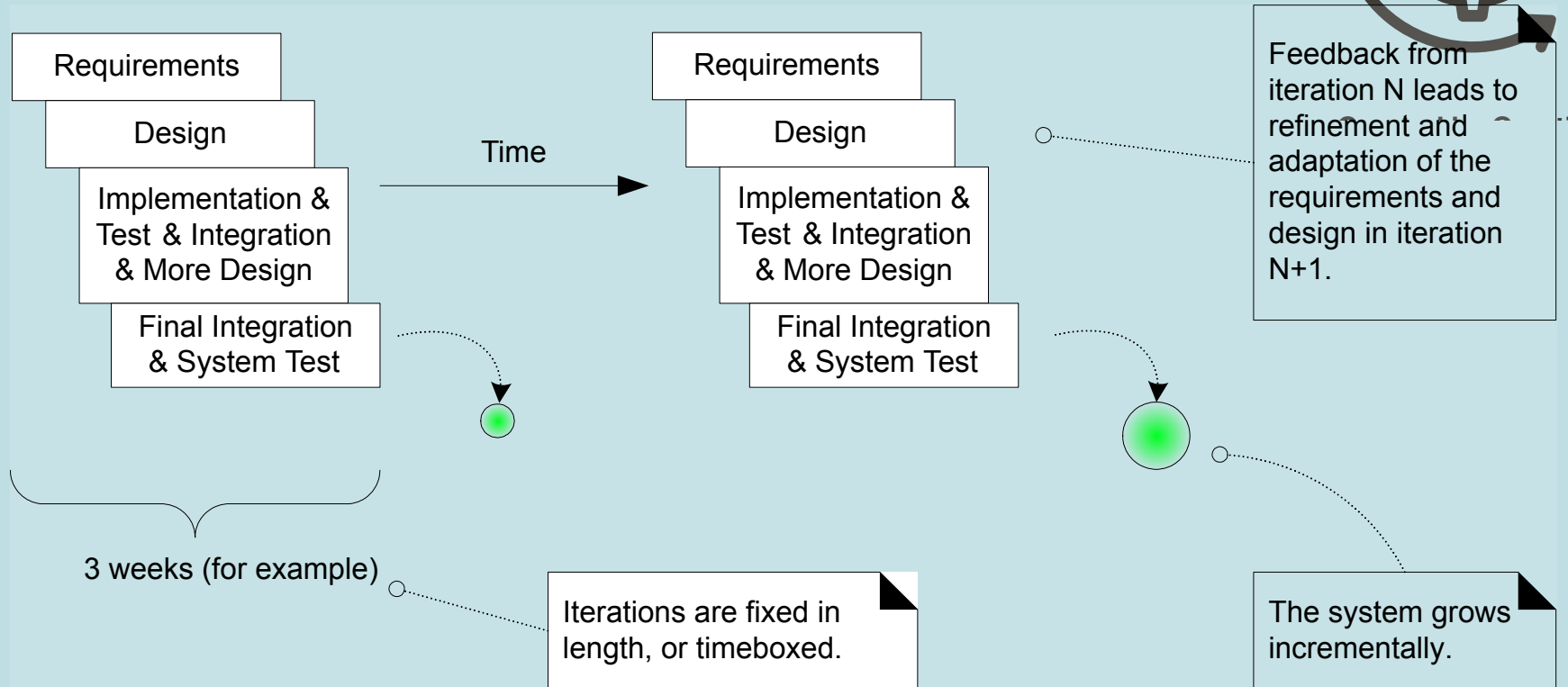
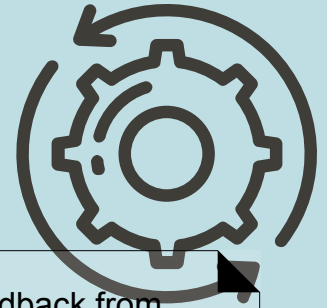
1. Dette technique
2. Rappel architecture en couches
3. Diagrammes d'état
4. Diagrammes d'activité
5. Processus Unifié (développement itératif et incrémental) ← 13.23m
6. GRASP dans les GoF

# PROCESSUS UNIFIÉ



- Souple et ouvert – peut accommoder l'Extrême Programming (XP), Scrum, etc.
- Développement piloté par les tests
- Processus *itératif* et *évolutif*

# ITÉRATIF ET ÉVOLUTIF

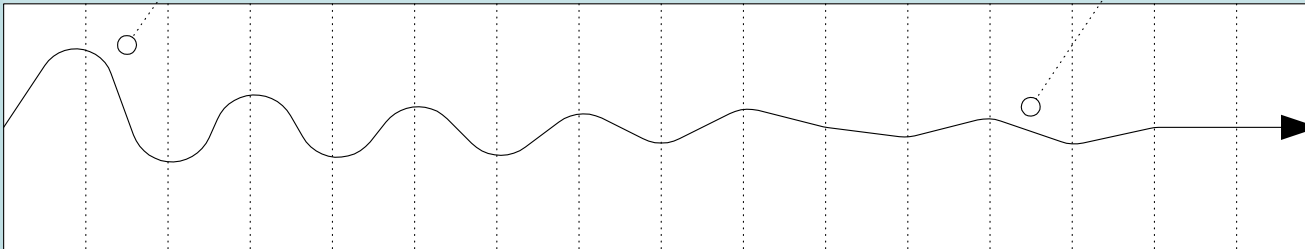


# STABILISATION



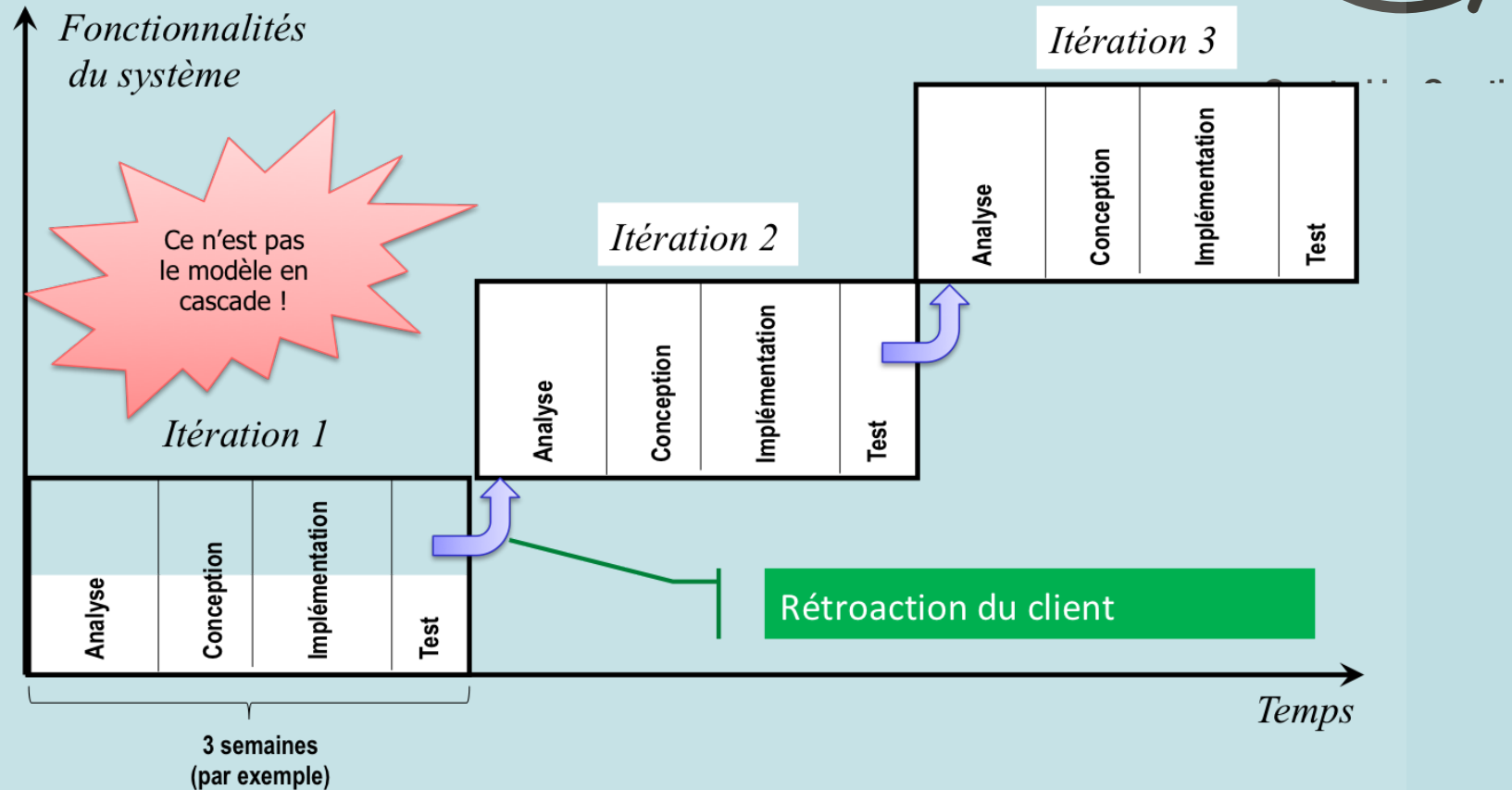
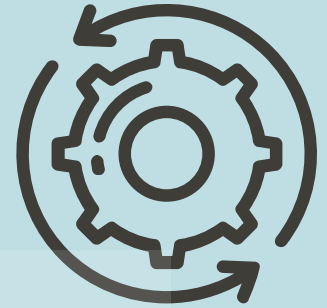
Les premières itérations sont loin de la «bonne voie». Grâce au feed-back et à l'adaptation, le système converge vers les spécifications et la conception les plus appropriées.

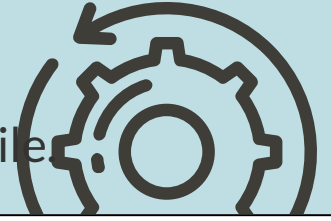
Lors des itérations ultérieures, un changement significatif des spécifications est rare mais peut survenir. De telles modifications tardives peuvent procurer à une entreprise un avantage concurrentiel.



une itération : conception,  
implémentation, intégration et tests

# ITÉRATIF ET ÉVOLUTIF





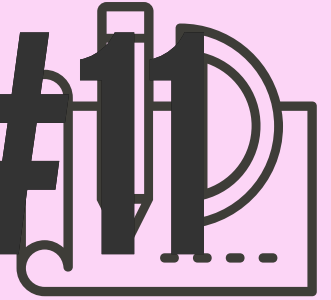
Écrire des spécifications correctes et complètes n'est pas facile.

Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit





# LOG210 SÉANCE #11



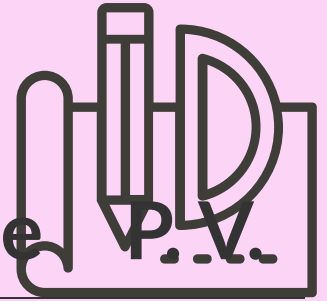
## OBJECTIF DE LA SÉANCE

1. Dette technique
2. Rappel architecture en couches
3. Diagrammes d'état
4. Diagrammes d'activité
5. Processus Unifié (développement itératif et incrémental)
6. GRASP dans les GoF ← 7.27m

# GRASP DANS LES GOLF

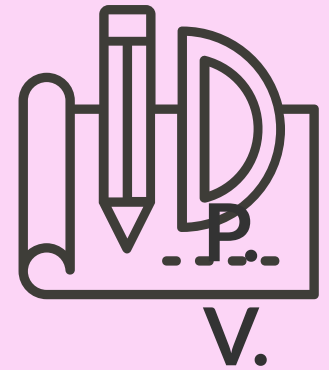
## Rapport technique

# TABLE 7: GRASP DANS LES GOF



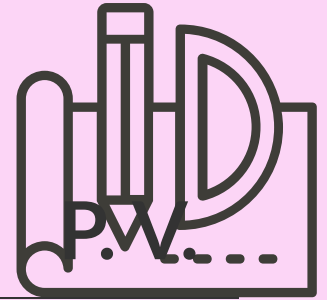
Pattern GoF	Indir.	Poly.	Fab. Pure	P.V.
Abstract factory	✓	✓	✓	✓
Builder	✓	✓	✓	✓
Factory method	✓	✓		✓
Prototype	✓	✓		✓
Singleton				
Adapter	✓	✓	✓	✓
Bridge	✓	✓	✓	✓

# TABLE 7: GRASP DANS LES GOF



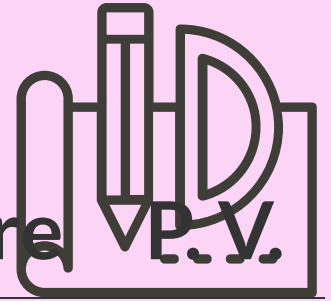
Pattern GoF	Indir.	Poly.	Fab. Pure	P. V.
Composite	✓	✓		✓
Decorator	✓	✓		✓
Facade	✓		✓	✓
Flyweight	✓	✓	✓	✓
Proxy	✓	✓	✓	✓
Chain of responsibility	✓	✓	✓	✓

# TABLE 7: GRASP DANS LES GOF



Pattern GoF	Indir.	Poly.	Fab. Pure	P.W. ...
Command	✓	✓	✓	✓
Interpreter	✓	✓	✓	✓
Iterator	✓	✓	✓	✓
Mediator	✓	✓	✓	✓
Memento	✓		✓	✓
Observer	✓	✓	✓	✓
State	✓	✓		✓

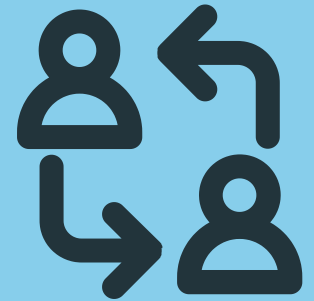
# TABLE 7: GRASP DANS LES GOF



Pattern GoF	Indir.	Poly.	Fab. Pure	P.V.
Strategy	✓	✓		✓
Template method	✓	✓		✓
Visitor	✓	✓	✓	✓

# SÉANCE #11

## RÉTROACTION: PAGE D'UNE MINUTE



Created by Prithvi  
from the Noun Project

1. Quels sont les deux [trois, quatre, cinq] plus importants [utiles, significatives, surprenantes, dérangementantes] choses que vous avez apprises au cours de cette session?
2. Quelle (s) question (s) reste (s) en tête dans votre esprit?
3. Y a-t-il quelque chose que tu n'as pas compris?

<https://1drv.ms/u/s!An6-F73ulxAOhVyiCB46jTeINVLs>

