



**Architecture et Agilité :
réconcilier les frères ennemis**

Jean-Philippe Gouigoux

11 Octobre



pôle **A**rchitecture / **F**ormation / **I**nnovation

search://gouigoux

Blog de JP Gouigoux

Retours d'expérience .NET d'un architecte logiciel sur des technologies "just get the job done"



Comprendre MongoDB depuis C#

Posté le 20/08/2012 by JP Gouigoux

Dans la lignée des articles du type "je ne comprends rien, mais je partage quand même" (voir les articles récents sur Async), voici quelques notes sur mon premier contact avec MongoDB. C'est une sorte de test, pour voir si la courbe d'apprentissage peut être rendue plus facile en montrant toutes les étapes de la prise de contact (y compris et surtout les erreurs et incompréhensions) plutôt qu'en expliquant directement le résultat correct.

Du coup, même avertissement que précédemment : ce que vous voyez ci-dessous n'est peut-être pas la bonne façon de faire. Ne vous en inspirez pas pour écrire du vrai code...

Installation

Histoire de faciliter la vie aux très grands débutants en MongoDB, comme moi, je vais détailler fortement les étapes de l'installation d'une configuration fonctionnelle en C#, sous Windows.

Abonnements



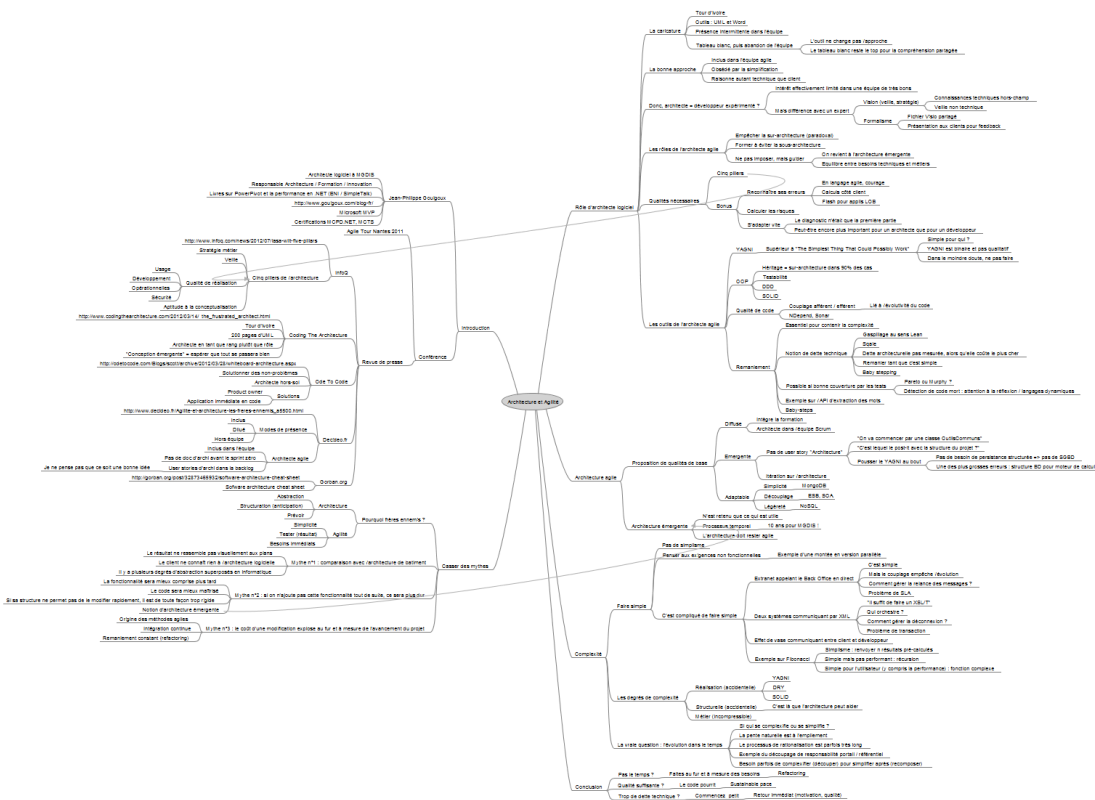
Mots-clés

Accelerator Agile
Apache Architecture ASP.NET Async
Azure
Bamboo
Cloud IaaS PaaS
GreenIT Hadoop
HTML5 Java JS
Lambda Lambda Live
Mono Mono IDE
OData
Performance
SHA-1 SharePoint
VisualStudio
VS2010 XSL/FO



AVANT DE COMMENCER

Mind Map de la session



Quoi de neuf ?



InfoQ.com : 5 piliers de l'architecture

- Stratégie métier
- Veille
- Qualité réalisation
- Aptitude à la conceptualisation
- Dynamique d'équipe



- Tour d'ivoire
- 200 pages d'UML
- Architecte en tant que rang plutôt que rôle
- « Conception émergente »
= espérer que tout se passera bien

L'architecte frustré



<< Photos from my QCon London 2012 tutorial | Home | Moving fast requ

The frustrated architect

Giant leaps or deep turmoil?

http://www.codingthearchitecture.com/2012/03/14/the_frustrated_architect.html

Architecture de tableau blanc

- Solutionner des non-problèmes
- Architecte hors-sol
- Solutions
 - Product owner
 - Application immédiate



<http://odetocode.com/Blogs/scott/archive/2012/03/28/whiteboard-architecture.aspx>

Joli titre...

- Trois modes : inclus, dilué, hors équipe
- Architecte agile
 - Inclus dans l'équipe
 - Pas de doc d'architecture avant le sprint zéro
 - User stories d'architecture dans la backlog

Decideo.fr

AGILITÉ ET ARCHITECTURE : LES FRÈRES ENNEMIS ?
L'agilité sonne-t-elle la fin de l'architecture ?

CHRISTOPHE COSNEFROY, NEOXIA
JEUDI 27 SEPTEMBRE 2012

Les pratiques agiles sont de plus en plus présentes au sein des DSI. Ce développement redéfinit le rôle de l'architecte dans les projets et plus généralement au sein de la DSI. Néanmoins les architectes se doivent d'avoir une vision à long terme en accord avec la stratégie de l'entreprise contrairement aux équipes agiles qui ont une vision à court terme axée sur la livraison d'un produit. Divers positionnements existent dans l'organisation : "inclus", "dilué" ou "en dehors de l'équipe". Pour avoir vécu personnellement vécu deux de ces situations, aucune ne m'a convaincu.

Expérience "Les opposants" : opposition entre développement agile et architecture

Deux visions s'opposent.

La première du département d'architecture est une vision long terme. Le département livre des documents d'architecture plusieurs mois en amont des développements, ces documents traduisant la stratégie de l'entreprise. Les inconvénients majeurs sont la prise en compte de besoins métiers non matures, la livraison de fonctionnalités inutiles et une documentation peu abordable.

La seconde vision, de l'équipe de développement, est centrée sur la livraison d'un produit, à court terme, au travers de la production de code.

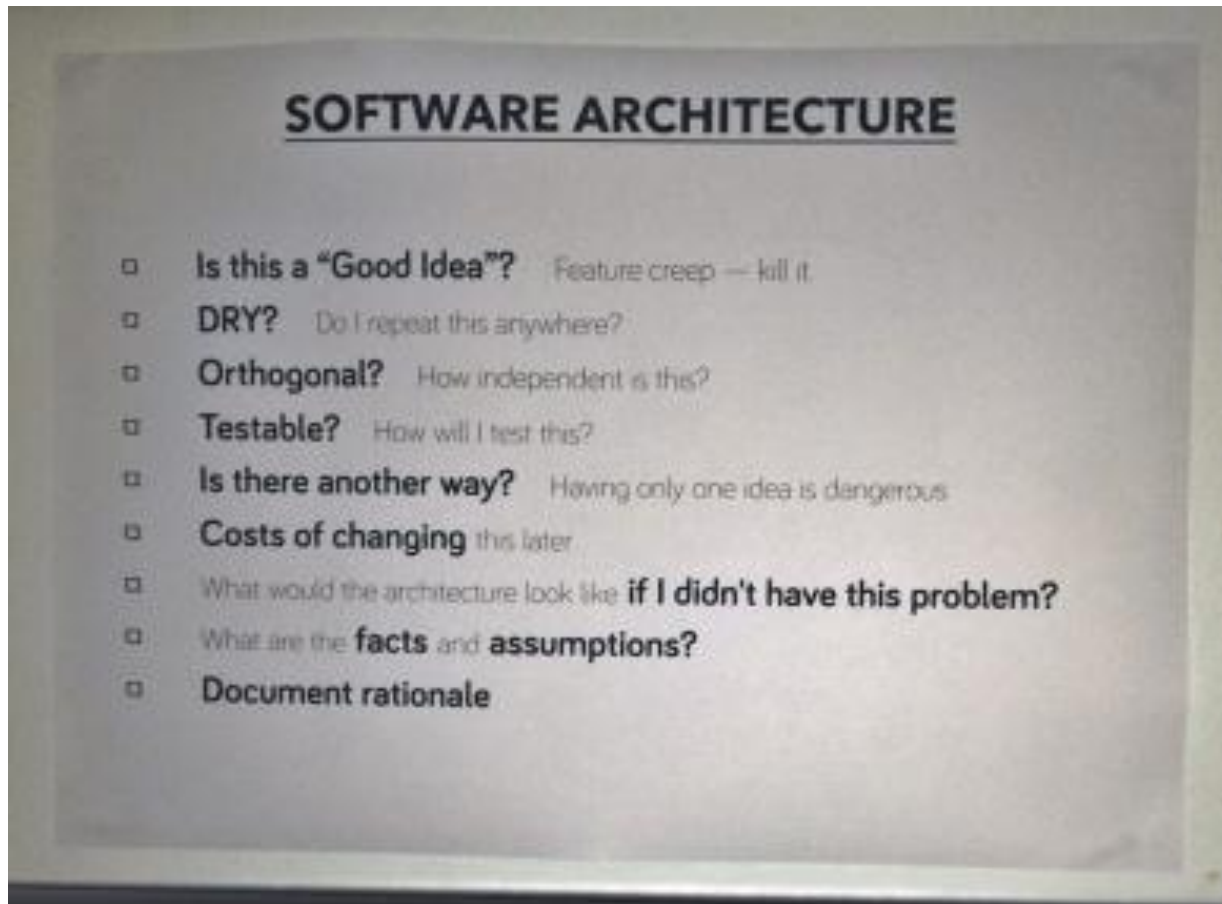
Cette opposition a pour avantage de clarifier les rôles de chacun. Les développeurs restent maîtres de leurs choix et leur objectif primordial reste la livraison du produit.



Christophe Cosnefroy, Consultant Sénior, chez Neoxia

http://www.decideo.fr/Agilite-et-architecture-les-freres-ennemis_a5500.html

L'inévitable « cheat sheet »



Gorban.org
INDIE SOFTWARE BUSINESS AND COCOA DEVELOPMENT

October 4, 2012 33 notes & 42 Comments and 668 Reactions

Software Architecture cheat sheet

For the past several weeks I've been focusing my efforts on learning how to approach software architecture. Despite my experience in developing several applications, I wanted to read and learn more about this to do a better job in the future, for our upcoming project.

I've read some articles and a couple of books on this topic. My goal was to extract relevant pieces of knowledge with the final goal of producing one sheet of paper with the most important points. I wanted to stick it onto the wall behind my screen, so that I could refer to it with only a glance. It had to have the most important questions that I should ask myself before committing to anything important during software design.

Today, I finally condensed all I wanted to write, prepared this page, printed and glued it to the wall:

About the author
Jacob Gorban founded Apparent Software, a Mac indie software company. Along with writing code, he loves the business side of things, plays guitars and listens to progressive rock.

Jacob can be found on Twitter and on App.net as @apparentsoft.

Ask me anything

Blog Tools

Search

<http://gorban.org/post/32873465932/software-architecture-cheat-sheet>

Pourquoi « frères ennemis » ?

- Architecture =

- Abstraire
- Structurer
- Prévoir



- Agilité =

- Simplicité (XP, Lean)
- Prééminence des tests (TDD)
- Besoins immédiats (YAGNI)

Buts de la session

- Détruire des mythes
- Rôle et outils de l'architecte logiciel
- Qualités d'une « architecture agile »
- Gestion de la complexité

DETRUIRE DES MYTHES

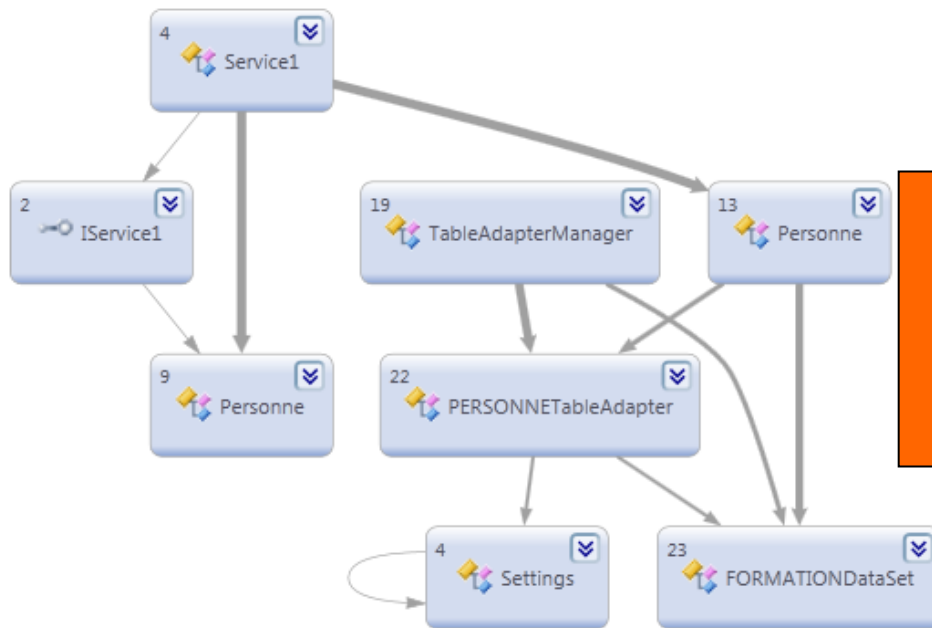
Mythe n°1

- « Architecture logicielle »

Architecture en bâtiment

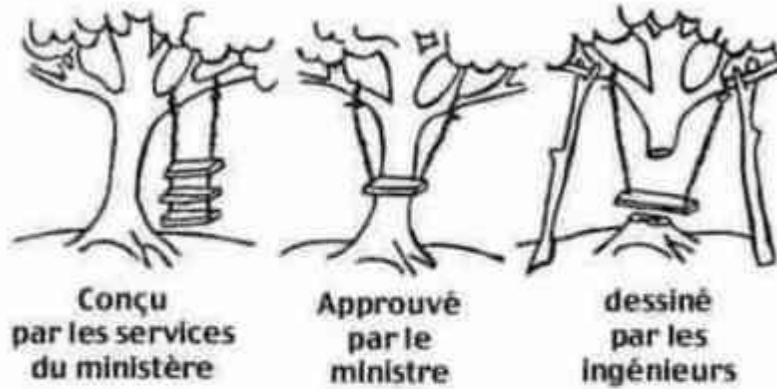


« Architecture » logicielle ?

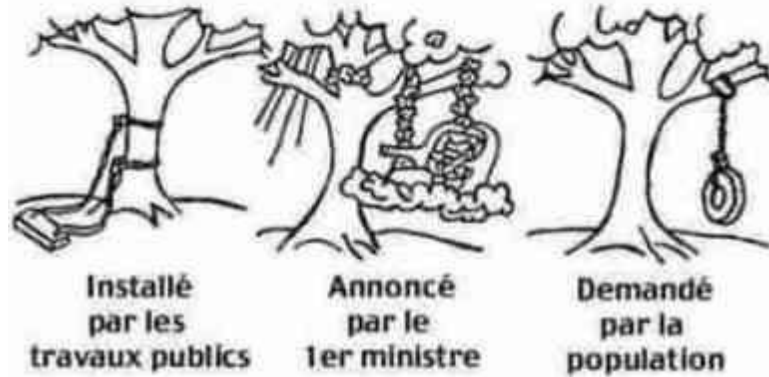


Pourquoi ?

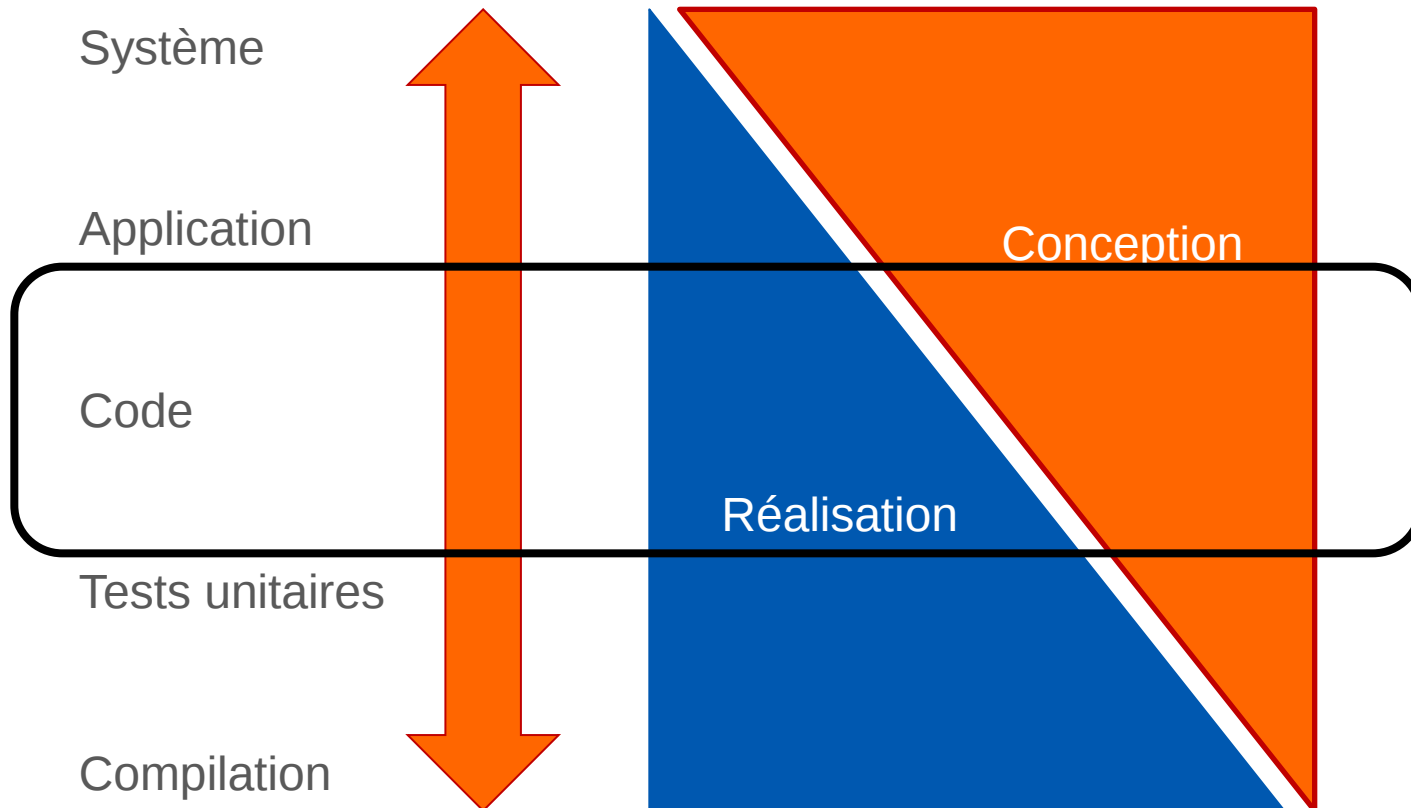
Projet du gouvernement



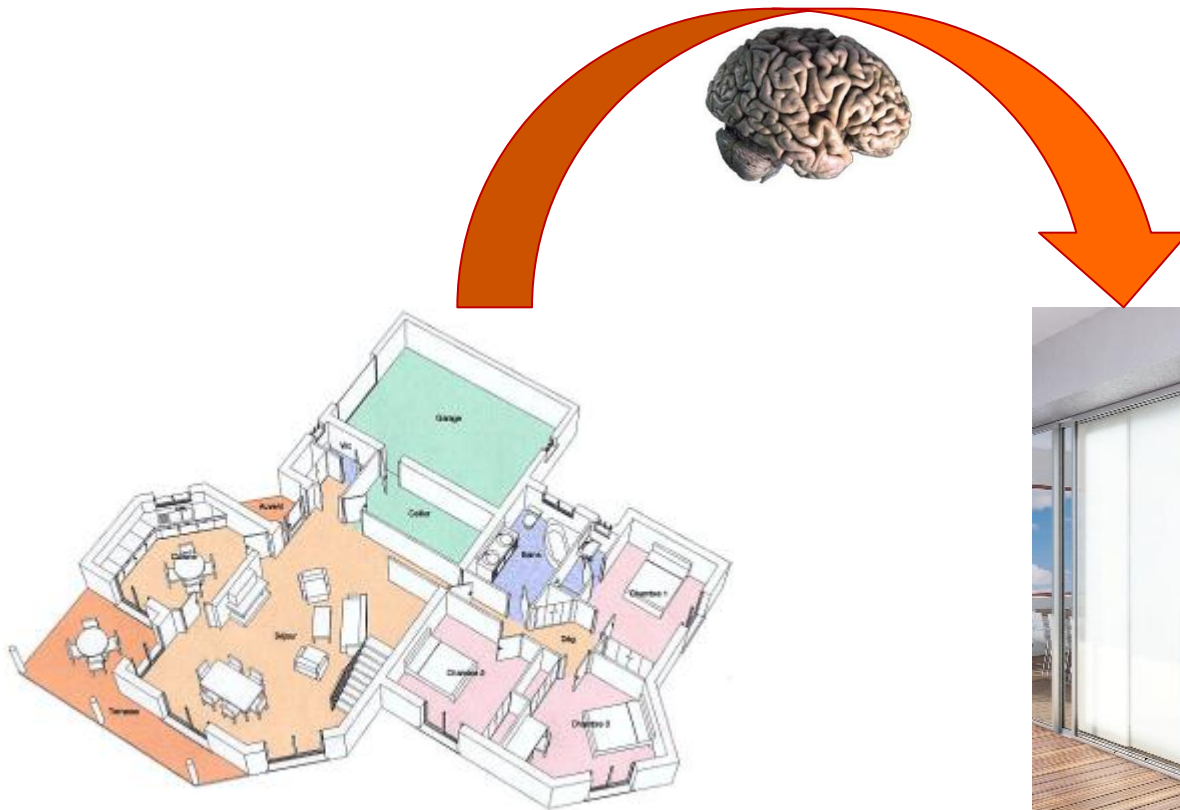
www.jemeclate.com



Le développeur n'est pas un maçon logiciel

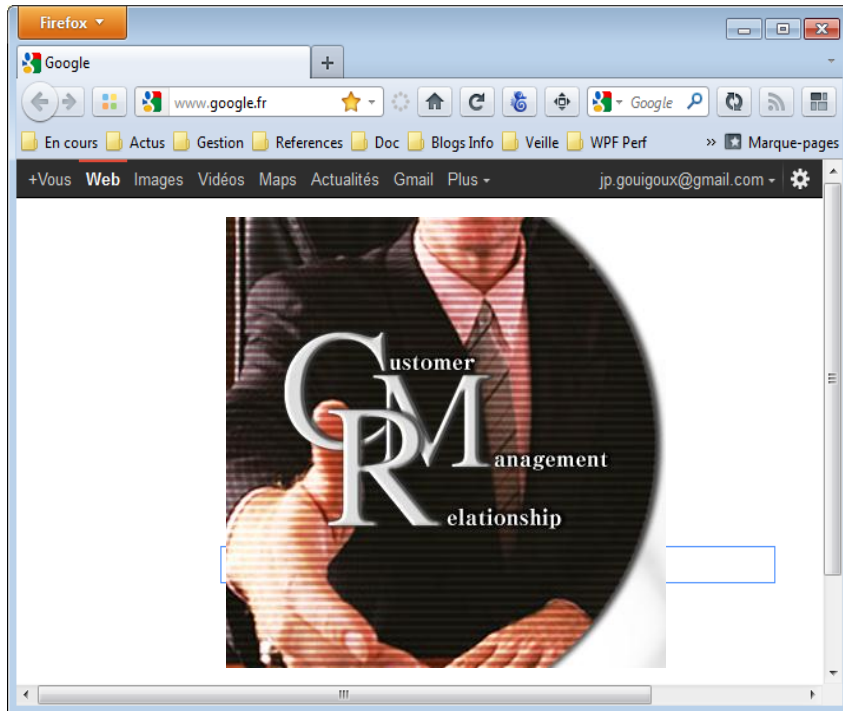


Abstraction





Pareil en info ?



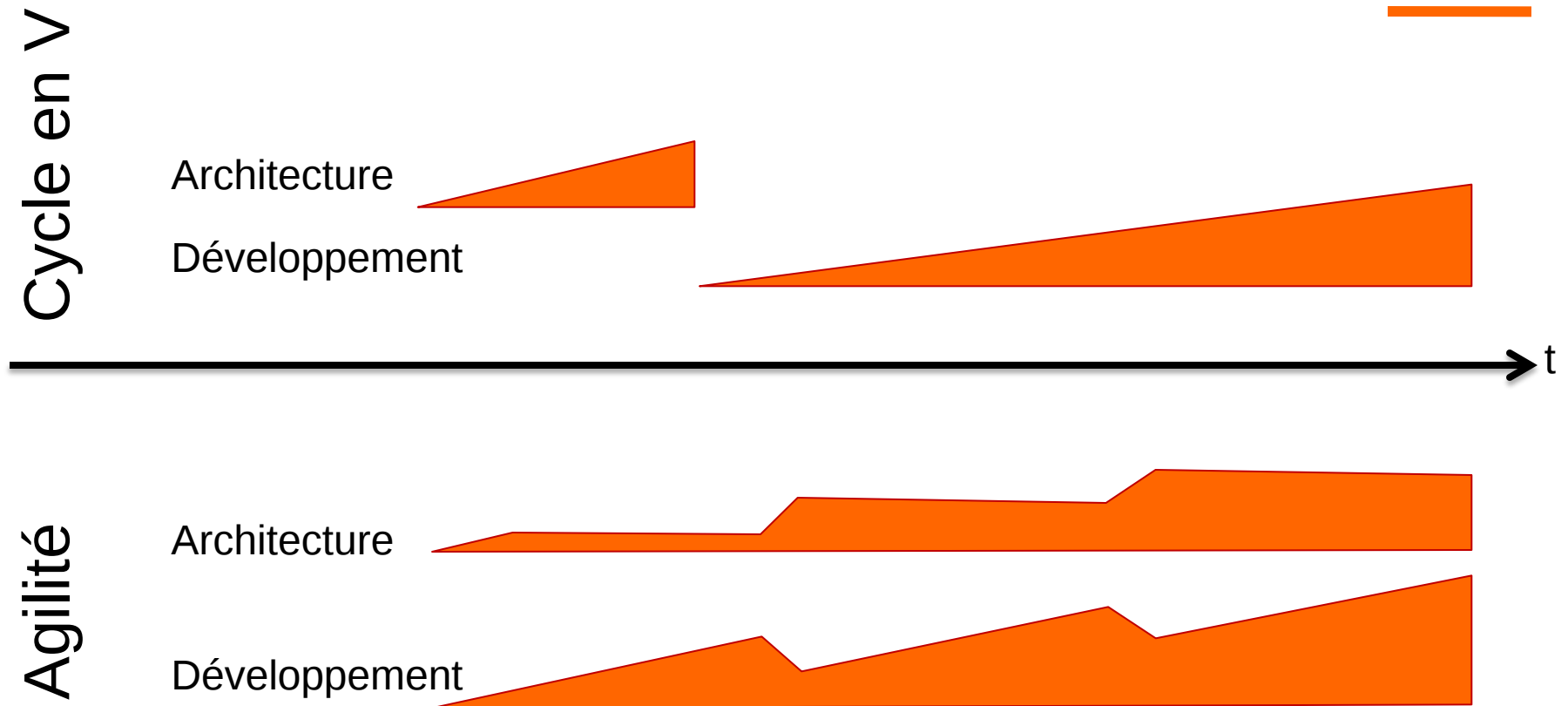
Oui, mais...



Mythe n°2

- « Si on n'inclut pas cette fonctionnalité tout de suite, on aura du mal à la rajouter »
 - **Faux**, car plus tard, la fonctionnalité voulue sera mieux comprise
 - **Faux**, car plus tard, on aura acquis plus de maîtrise du code existant
 - **Faux**, car si l'architecture ne permet pas cette modification, elle était de toute façon trop rigide

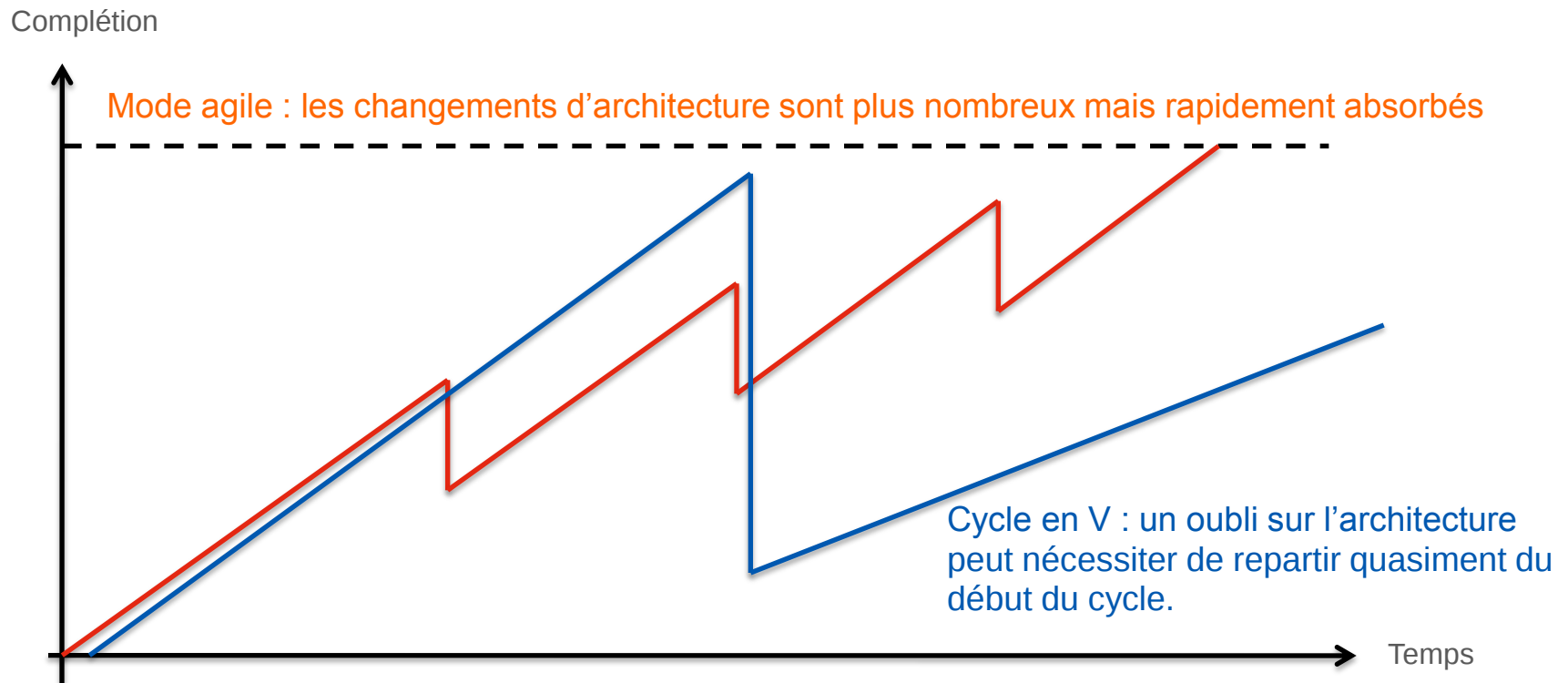
Concept d' « architecture émergente »



Mythe n°3

- « Le coût d'une modification augmente exponentiellement avec la progression dans le projet »
 - **Faux** en mode agile : l'intégration continue garantit une livraison rapide (même si coût en amont)
 - **Faux** en mode agile : le remaniement constant de l'application (XP) fait qu'un changement d'architecture ne prendra pas plus de temps sur un sprint que sur un autre

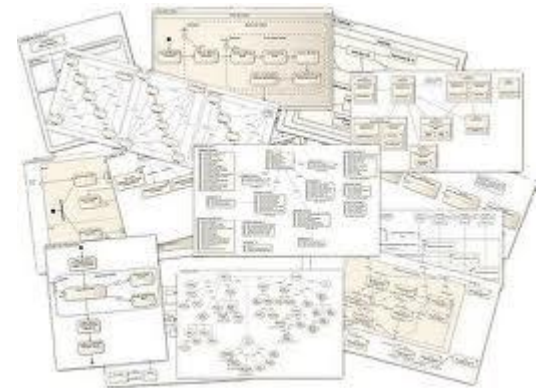
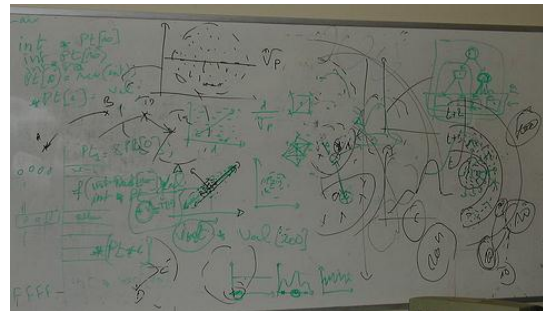
Corollaire sur le temps total d'un projet



LE ROLE D'ARCHITECTE LOGICIEL

La caricature

- Moquerie de bon ton
 - Tour d'ivoire
 - Communication par tableau blanc
 - Programmation UML



La part de réel

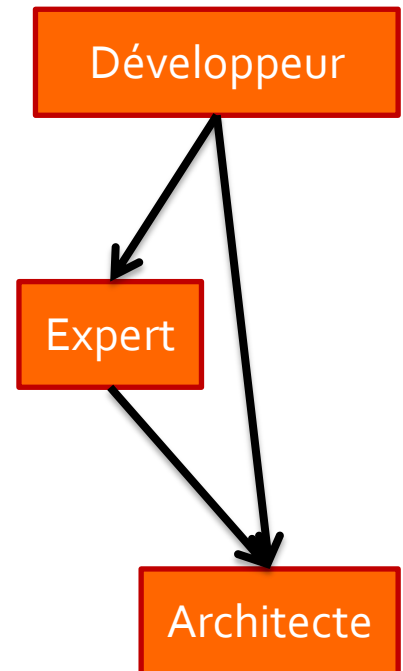
- Beaucoup d'échecs de projets IT à cause d'une sur-architecture par rapport aux besoins
- Architecte souvent vu comme un rang plutôt qu'un rôle

Bon chasseur, mauvais chasseur

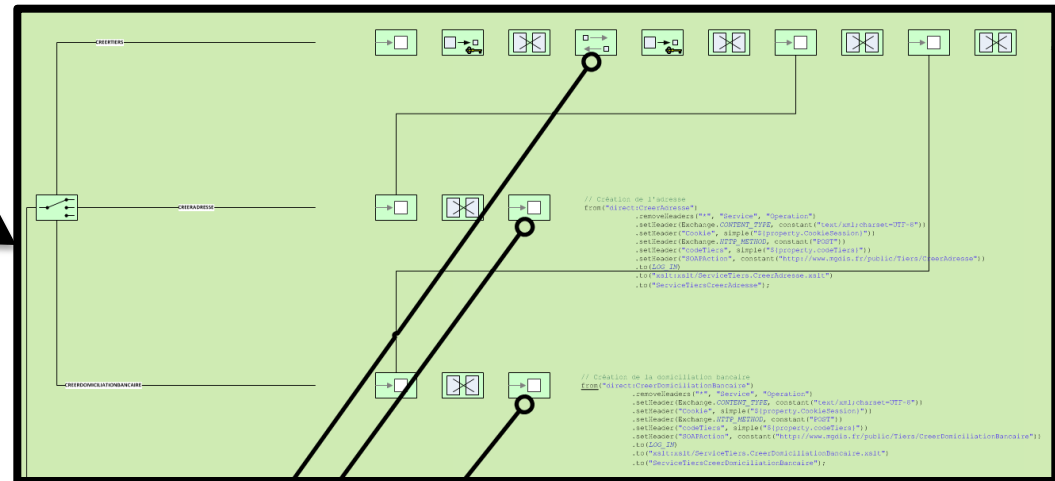
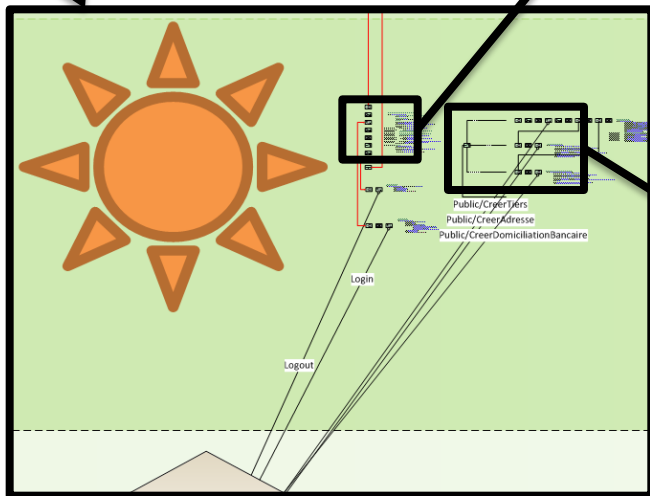
- Niveau conceptuel
- Rôle différent du développeur
- Ne pense pas dans l'immédiat

Un développeur expérimenté ?

- Intérêt architecte limité en fonction de l'équipe
- Différent de l'expert
 - Vision
 - Connaissances techniques hors-champ
 - Veille non technique
 - Formalisme
 - Mise en forme pour partage efficace
 - Lien client pour feedback







Rôle de l'architecte en milieu agile

- Empêcher la sur-architecture (paradoxal)
- Diffuser les bonnes pratiques
- Ne pas imposer, mais guider
 - Architecture émergente
 - Equilibre technique / métier

Qualités de l'architecte (InfoQ)

- Stratégie métier
- Veille
- Qualité réalisation
 - Usage
 - Développement
 - Opérationelles
 - Sécurité
- Aptitude à la conceptualisation
- Dynamique d'équipe



Qualités de l'architecte (ajouts perso)

- Reconnaître ses erreurs, même énormes
 - Calculs côté client
 - Flash pour application LOB
 - 80 tables au lieu d'un fichier XML
- Calculer les risques (formalisme)
- S'adapter vite
 - Le diagnostic ne suffit pas (et dévalorise le métier)
 - Agilité peut-être encore plus importante pour l'archi



Les outils de l'architecte (1)

- YAGNI (supérieur à DTSTTCPW)
- OOP
 - Héritage = sur-architecture dans 90% des cas
 - Utiliser plutôt SOLID
 - Domain Driven Design
 - Coder pour la testabilité



SOLID : Top
investissement !!!

Les outils de l'architecte (2)

- Qualité de code
 - Couplage afférent / efférent
 - NDepend, Sonar, FXCop
- Remaniement
 - Contenir la complexité
 - Dette technique (gaspillage Lean, Sqale, dette archi)
 - Couverture de code : Pareto ou Murphy ?
 - Nettoyage de code mort

Mise en place du filet de sécurité

```
[TestFixture]
public class TestExtractionMots
{
    [Test]
    public void ExtractionChaineVideMotsCourts()
    {
        Assert.AreEqual(0, Program.ExtraireMotsCourts(string.Empty).Count);
    }

    [Test]
    public void ExtractionSimpleMotsCourts()
    {
        Assert.AreEqual(4, Program.ExtraireMotsCourts("De la terre à la lune").Count);
    }

    [Test]
    public void ExtractionChaineVideMotsLongs()
    {
        Assert.AreEqual(0, Program.ExtraireMotsLongs(string.Empty).Count);
    }

    [Test]
    public void ExtractionSimpleMotsLongs()
    {
        Assert.AreEqual(1, Program.ExtraireMotsLongs("De la commande à la lune").Count);
    }
}
```

Etape 1

```
public static List<string> ExtraireMotsCourts(string Texte)
{
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (Mot.Length < 3) → if (EstUnMotCourt(Mot))
            Liste.Add(Mot);
    return Liste;
}

public static List<string> ExtraireMotsLongs(string Texte)
{
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (Mot.Length > 5) → if (EstUnMotLong(Mot))
            Liste.Add(Mot);
    return Liste;
}

public static bool EstUnMotCourt(string Mot)
{
    return Mot.Length < 3;
}

public static bool EstUnMotLong(string Mot)
{
    return Mot.Length > 5;
}
```

Etape 2

```
public static List<string> ExtraireMotsCourts(string Texte)
{
    TesteurMot Critere = EstUnMotCourt;
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (EstUnMotCourt(Mot)) → if (Critere(Mot))
            Liste.Add(Mot);
    return Liste;
}

public static List<string> ExtraireMotsLongs(string Texte)
{
    TesteurMot Critere = EstUnMotLong;
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (EstUnMotLong(Mot)) → if (Critere(Mot))
            Liste.Add(Mot);
    return Liste;
}

public delegate bool TesteurMot(string Mot);
public static bool EstUnMotCourt(string Mot)
{
    return Mot.Length < 3;
}

public static bool EstUnMotLong(string Mot)
{
    return Mot.Length > 5;
}
11/10/2012 }
```

Etape 3

```
public static List<string> ExtraireMotsCourts (string Texte)
{
    TesteurMot Critere = EstUnMotCourt;
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (Critere(Mot))
            Liste.Add(Mot);
    return Liste;
}

public static List<string> ExtraireMotsLongs (string Texte)
{
    TesteurMot Critere = EstUnMotLong;
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (Critere(Mot))
            Liste.Add(Mot);
    return Liste;
}

public delegate bool TesteurMot(string Mot);
public static bool EstUnMotCourt (string Mot)
{
    return Mot.Length < 3;
}

public static bool EstUnMotLong (string Mot)
{
    return Mot.Length > 5;
}
```

Etape 4

```
public static List<string> ExtraireMotsCourts(string Texte)
{
    TesteurMot Critere = EstUnMotCourt;
    return ExtraireMots(Texte, Critere);
}
public static List<string> ExtraireMotsLongs(string Texte)
{
    TesteurMot Critere = EstUnMotLong;
    return ExtraireMots(Texte, Critere);
}
public static List<string> ExtraireMots(string Texte, TesteurMot Critere)
{
    List<string> Liste = new List<string>();
    foreach (string Mot in Texte.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries))
        if (Critere(Mot))
            Liste.Add(Mot);
    return Liste;
}
public delegate bool TesteurMot(string Mot);
public static bool EstUnMotCourt(string Mot)
{
    return Mot.Length < 3;
}
public static bool EstUnMotLong(string Mot)
{
    return Mot.Length > 5;
}
```

Etape 5

```

public static List<string> ExtraireMotsCourts (string Texte)
{
    return ExtraireMots (Texte, EstUnMotCourt);
}
public static List<string> ExtraireMotsLongs (string Texte)
{
    return ExtraireMots (Texte, EstUnMotLong);
}
public static List<string> ExtraireMots (string Texte, TesteurMot Critere)
{
    return new List<string> (Texte.Split (new char[] { ' ' },
        StringSplitOptions.RemoveEmptyEntries)).FindAll (Critere);
}
return Liste;
}
public delegate bool TesteurMot (string Mot);
public static bool EstUnMotCourt (string Mot)
{
    return Mot.Length < 3;
}
public static bool EstUnMotLong (string Mot)
{
    return Mot.Length > 5;
}

```

Etape 6

```
public static List<string> ExtraireMotsCourts(string Texte)
{
    return ExtraireMots(Texte, EstUnMotCourt);
}
public static List<string> ExtraireMotsLongs(string Texte)
{
    return ExtraireMots(Texte, EstUnMotLong);
}
public delegate bool TesteurMot(string Mot);
public static List<string> ExtraireMots(string Texte, TesteurMot Critere)
{
    return new List<string>(Texte.Split(new char[] { ' ' },
        StringSplitOptions.RemoveEmptyEntries)).FindAll(Critere);
}

private delegate bool TesteurMot(string Mot);
public static bool EstUnMotCourt(string Mot)
{
    return Mot.Length < 3;
}
public static bool EstUnMotLong(string Mot)
{
    return Mot.Length > 5;
}
```

Plus simple ou pas ?

- Plus long au début, puis plus court
- Plus dur à lire pour un débutant
- On a fait apparaître des « caractéristiques désirables du code » (SRP, en particulier)

Atteindre l'étape « architecture »

```
public delegate bool TesteurMot(string Mot);  
public static List<string> ExtraireMots(string Texte, TesteurMot Critere)  
{  
    return new List<string>(Texte.Split(new char[] { ' ' },  
        StringSplitOptions.RemoveEmptyEntries).FindAll(Critere);  
}  
  
private static bool IsMotCourt(string Mot)  
{  
    return Mot.Length < 5;  
}  
private static bool IsMotLong(string Mot)  
{  
    return Mot.Length > 5;  
}
```

} DLL API

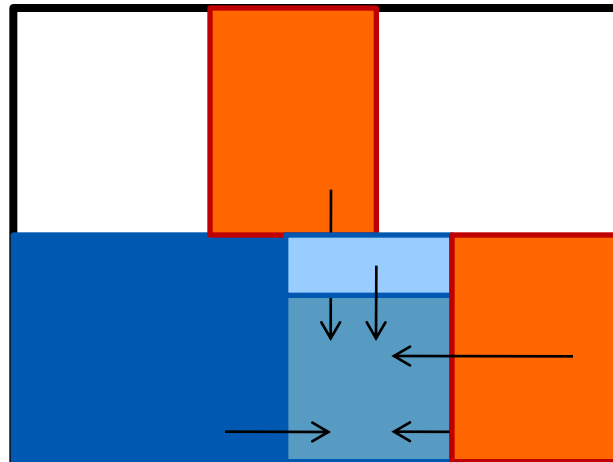
} DLL Métier

Notion complexe /
Architecture simple

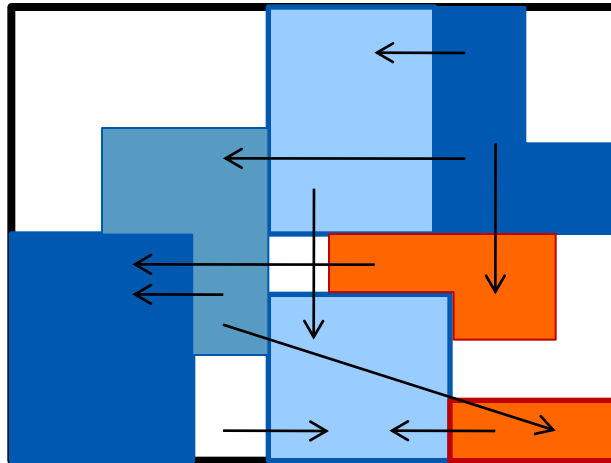
Nous sommes retombés de
nous-mêmes sur la notion
de délégué anonyme :

```
Program.ExtraireMots(ChaineTest, delegate(string Mot)  
{  
    return Mot.Contains("o");  
});
```

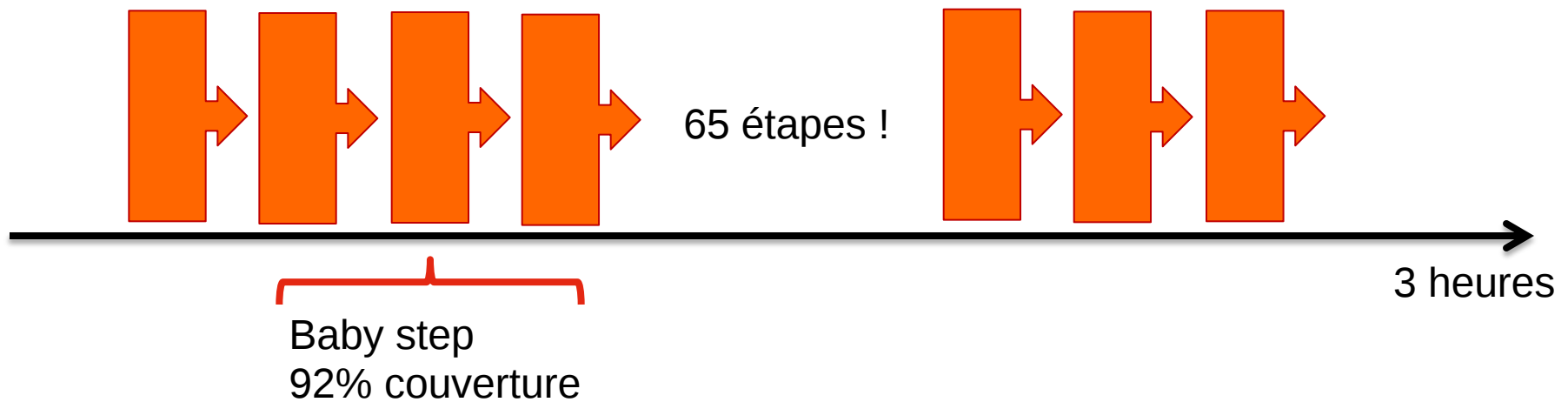
Récursion code simple / remaniement simple



Code complexe : bon courage !



Exemple vécu



ROI : immédiat !

Baby steps ?

- Tout petits pas itératifs
 - Correction simpliste
 - Tests unitaires
 - Retours sur erreurs



Tests unitaires

ARCHITECTURE AGILE ?

Proposition de propriétés de base

- Diffuse
 - Intègre la formation / diffusion des pratiques
 - Architecte inclus dans l'équipe SCRUM
- Emergente
 - Pas de user story « architecture »
 - Architecture itérative (même si long terme)
- Adaptable
 - Simplicité, découplage, légèreté (MongoDB, ESB, ...)

Approche Agile

Backlog

p3 8h
En tant qu'administrateur de la solution, je dois pouvoir sécuriser l'accès aux données par rapport à l'utilisateur, et empêcher les accès externes.

p3 3h
En tant qu'utilisateur, je dois pouvoir insérer une nouvelle personne ou supprimer une entrée, ou plusieurs en une interaction.

p3 5h
En tant qu'utilisateur, je souhaite pouvoir afficher beaucoup de données sans problème de performance, et j'accepte la proposition de pagination.

p4 1h
Mettre en place une icône personnalisée pour l'application.

Backlog de sprint 1 (remise 07/10)

A faire

p1 1h
En tant qu'utilisateur, je dois voir d'un coup d'œil le nom, le prénom d'une personne ainsi que son n° de sécurité sociale.

p1 1h
En tant qu'utilisateur, je veux que les données se chargent dès que je démarre l'application.

Développement

* p1 1h
En tant que client demandeur de l'application, je fournirai un jeu de 50 données au format CSV pour tests.

Validation

Fin (au sens Agile : livré + testé + documenté)

Backlog de sprint 2 (remise 14/10)

A faire

p2 1h
En tant qu'utilisateur, je souhaite voir l'âge de la personne, calculé en partant de l'année et du mois trouvés dans le n° de sécurité sociale.

p2 21h
En tant qu'utilisateur, je souhaite pouvoir modifier les données sans avoir à valider à chaque fois, et voir la modification directement dans la grille. Le scénario est détaillé graphiquement sur le document attaché (cf diapositive suivante).

Développement

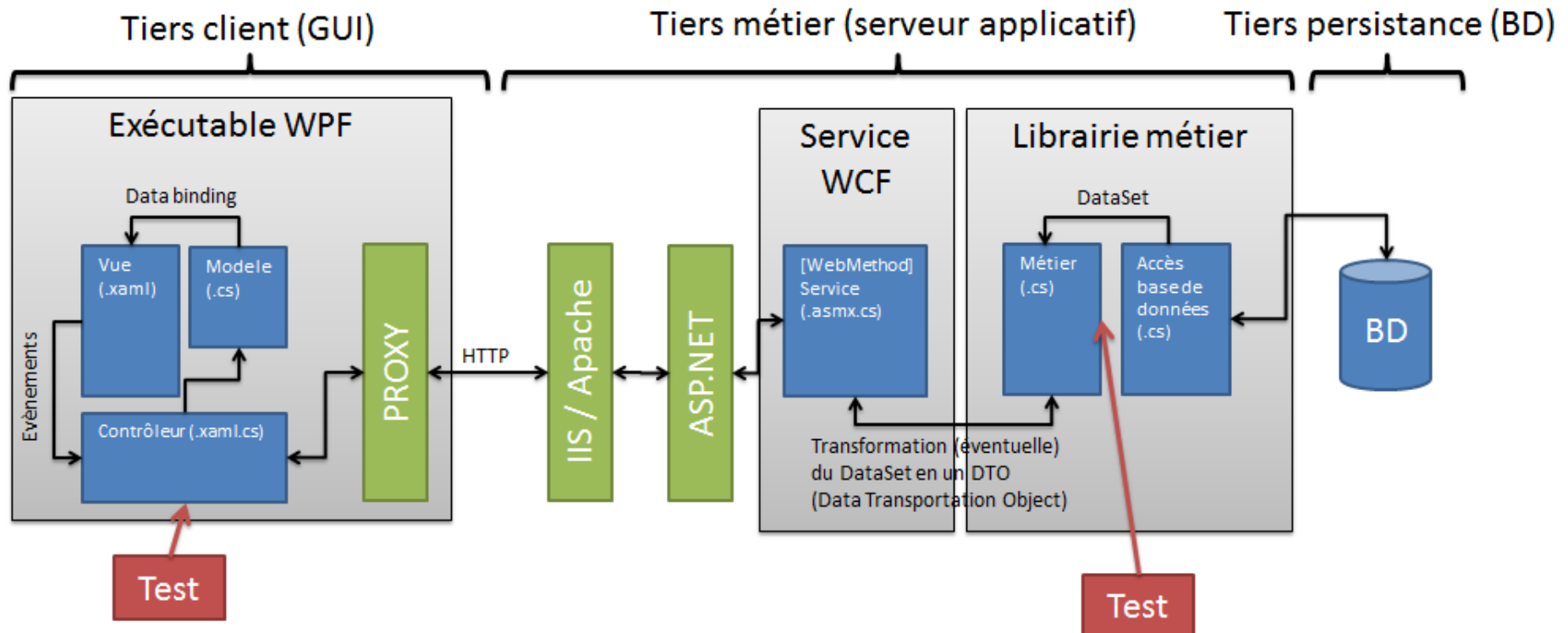
Validation

Fin (au sens Agile : livré + testé + documenté)

Deux retours immédiats

- « On va commencer par une classe Outils avec toutes les fonctions dont on a besoin »
 - Non ! (YAGNI)
- « C'est lequel le post-it avec la structure du projet? »
 - Solution et projets Visual Studio
 - Création de la fenêtre, du service, de la structure BD

Exemple : TP de formation .NET



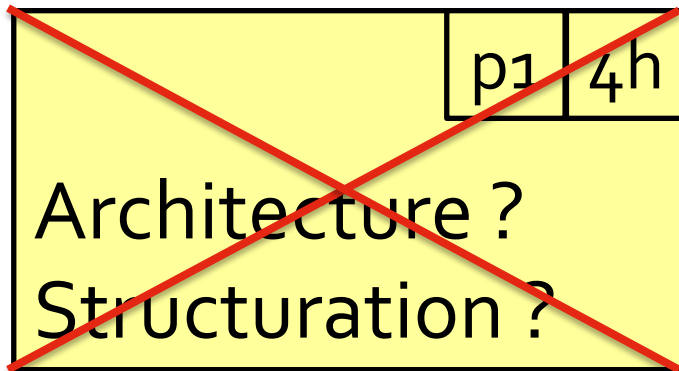
YAGNI

A faire

En tant qu'utilisateur, je dois voir d'un coup d'œil le nom, le prénom d'une personne ainsi que son n° de sécurité sociale.	p1	1h
En tant qu'utilisateur, je veux que les données se chargent dès que je démarre l'application.	p1	1h

- Conséquences
 - Pas de 3-tiers
 - Pas de service
 - Pas de base de données !
 - 1 solution avec 1 projet

Un post-it pour la structure initiale ?

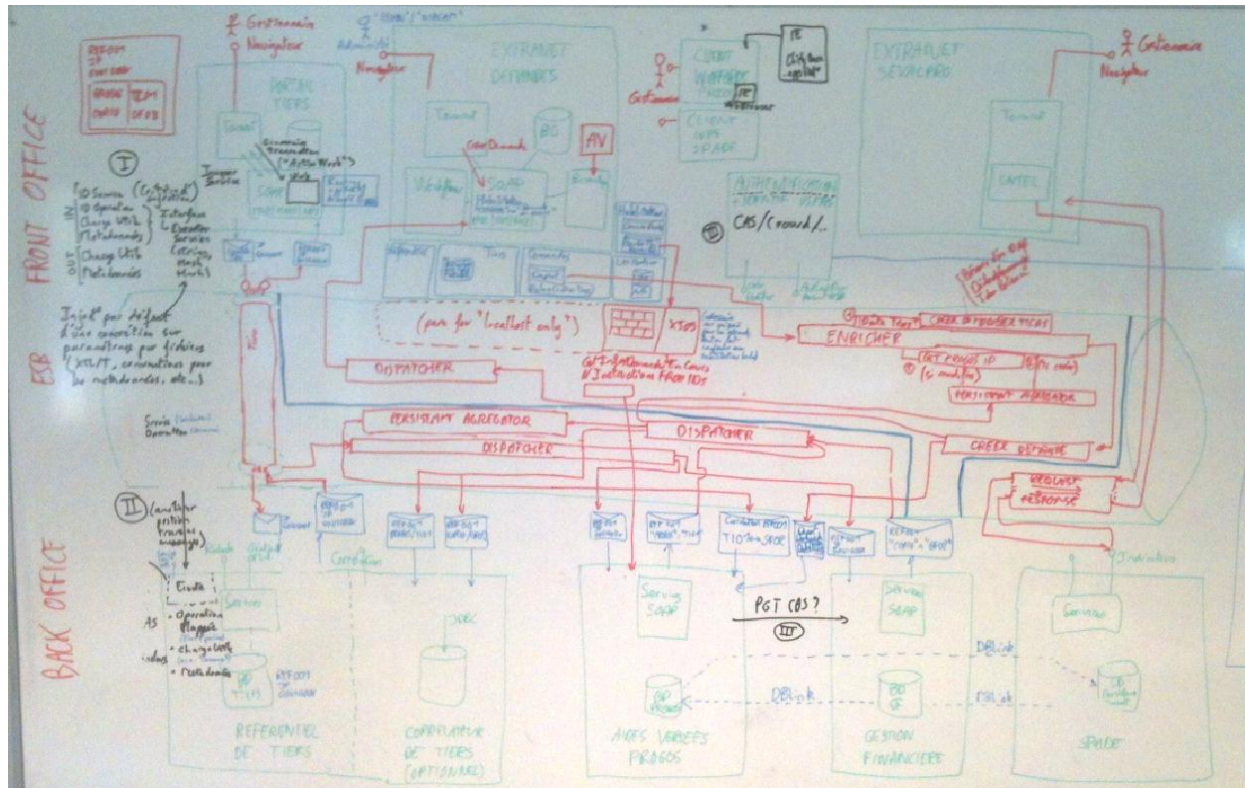


- Mais
 - Séparer GUI / métier
 - Contrats simples
 - Interface persistance

Architecture émergente

- « The frustrated architect » très critique
 - Hoping for the best
- L'émergence est facilitée par l'architecte
 - Vision stratégique des enjeux long terme
 - Connaissance des outils alternatifs
- Eviter la sur-architecture est l'objectif n°1

Quand je parle de long terme...



Mise en place d'une architecture SOA

- 2003 : web services
- 2005 : 100% SOAP
- 2007 : intégration autres applis
- 2009 : extranets
- 2010 : web services publics
- 2011 : REST
- 2012 : début urbanisation ESB
- 2015 (?) : architecture complètement SOA

REX sur une architecture émergente

- Ne prendre que ce qui est utile
 - L'équipe est convaincue
 - Un client est prêt à payer
- Etaler le risque dans le temps
- L'infrastructure doit être elle-même agile
 - ESB = souplesse
 - ESB propriétaire = menottes

GESTION DE LA COMPLEXITÉ

Faire simple

- Ne veut pas dire simpliste
- Les exigences non fonctionnelles
 - Robustesse
 - Sécurité
 - Montée en version parallélisable (besoin admin.)
 - Rarement exprimées dans la backlog
 - etc.

Fibonacci simpliste

```
public class Fibonacci
{
    public IEnumerable<int> Calculer()
    {
        yield return 1;
        yield return 1;
        yield return 2;
        yield return 3;
        yield return 5;
        yield return 8;
        yield return 13;
        yield return 21;
    }
}
```

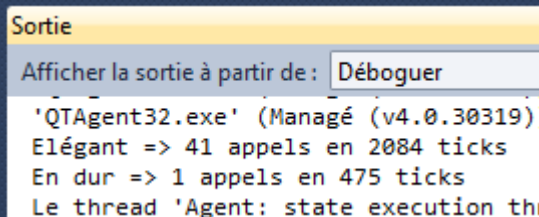
Fibonacci simple

```
public static class Math
{
    public static int Fibonacci(int Niveau)
    {
        if (Niveau < 2) return 1;
        return Fibonacci(Niveau - 1)
            + Fibonacci(Niveau - 2);
    }
}
```

... mais peu performant !

```
[TestClass]
public class TestCalcul
{
    [TestMethod]
    public void TestFiboElegant()
    {
        Calcul.ResetCompteur();
        Stopwatch Chrono = Stopwatch.StartNew();
        Assert.AreEqual(21, Calcul.FibonacciElegant(7));
        Chrono.Stop();
        Debug.WriteLine("Elégant => {0} appels en {1} ticks", Calcul.Compteur, Chrono.ElapsedTicks);
    }

    [TestMethod]
    public void TestFiboEnDur()
    {
        Calcul.ResetCompteur();
        Stopwatch Chrono = Stopwatch.StartNew();
        Assert.AreEqual(21, Calcul.FibonacciEnDur(7));
        Chrono.Stop();
        Debug.WriteLine("En dur => {0} appels en {1} ticks", Calcul.Compteur, Chrono.ElapsedTicks);
    }
}
```



Sortie

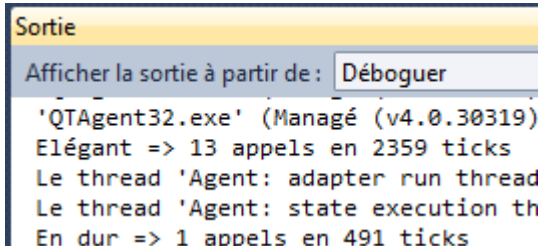
Afficher la sortie à partir de: Déboguer

'QTAgent32.exe' (Managé (v4.0.30319))
Elégant => 41 appels en 2084 ticks
En dur => 1 appels en 475 ticks
Le thread 'Agent: state execution thr

Fibonacci équilibré entre simple et performant

```
public static Dictionary<int, int> CacheFibo = new Dictionary<int, int>();

public static int FibonacciElegant(int Profondeur)
{
    _Compteur++;
    if (CacheFibo.ContainsKey(Profondeur)) return CacheFibo[Profondeur];
    int Resultat = 1;
    if (Profondeur > 1) Resultat = FibonacciElegant(Profondeur - 2) + FibonacciElegant(Profondeur - 1);
    CacheFibo.Add(Profondeur, Resultat);
    return Resultat;
}
```



Sortie

Afficher la sortie à partir de: Déboguer

'QTAgent32.exe' (Managé (v4.0.30319))
Elégant => 13 appels en 2359 ticks
Le thread 'Agent: adapter run thread'
Le thread 'Agent: state execution th'
En dur => 1 appels en 491 ticks

C'est compliqué de faire simple

- Extranet contacte Back Office
- Faire communiquer les serveurs ou transformer les messages ?
- Vases communicants entre
 - Complexité pour l'utilisateur
 - Complexité pour le développeur
- L'architecte doit équilibrer et maîtriser la complexité par des solutions hors-champ

Les degrés de complexité

- Réalisation (accidentelle)
 - YAGNI, DRY, SOLID
- Structurelle (accidentelle)
 - Ré-architecture
- Métier (incompressible)

Solution : aborder la complexité dans le temps

- SI qui se simplifie ou se complexifie ?
- Pente naturelle : empilement
- Rationalisation très longue
 - Comme pour le remaniement, parfois besoin de complexifier avant de re-décomposer
- Exemple : découplage d'un annuaire

JUST DO IT !

« Frères ennemis », vraiment ?

- Architecture =
 - Abstraire => simplifier
 - Structurer => assouplir
 - Prévoir => s'adapter
- Agilité =
 - Simplicité (XP, Lean)
 - Prééminence des tests (TDD)
 - Besoins immédiats (YAGNI)

Courage, notion fondamentale de l'agilité

- Pas le temps ?
 - Remaniement au fur et à mesure des besoins
- Qualité suffisante ?
 - Le code pourrit
 - Sustainable pace
- Trop de dette technique ?
 - Commencez petit
 - Assurez un retour immédiat (motivation, qualité)

Citation

- Comment pourrait-on faire du développement agile sur une architecture qui ne le serait pas ?
- Hugues Cornuaille, Certified Scrum Master

Questions... et peut-être même réponses !

