Alter Way

Livre blanc

Industrialisation PHP

Damien SEGUY
Jean-Marc FONTAINE

***Alter Way**, intégrateur Open Source de référence, accompagne ses clients au travers de quatre offres majeures : le conseil (Alter Way Consulting), la formation (Anaska, Alter Way Formation), l'intégration et le développement (Alter Way Solutions) et l'hébergement à valeur ajoutée (Nexen, Alter Way Hosting).*

Alter Way couvre les principaux besoins du systme d'information : gestion de contenu, ECM, e-commerce, CRM, business intelligence, applications métier, messagerie d'entreprise, infrastructures système et réseaux.

Son offre se caractérise par la dimension industrielle et l'excellence résultant, de la réunion sous la marque Alter Way d'équipes emblématiques dans leur spécialité.

Dirigé par ses fondateurs, Philippe Montargès et Véronique Torner, le groupe Alter Way apporte aux grands comptes, administrations, collectivités locales et PME/PMI, une réponse industrielle globale assurée par un interlocuteur unique.

Il fait valoir sa compétence distinctive du Libre ainsi qu'une capacité d'accompagnement et une pérennité comparables à celles des intégrateurs traditionnels.

Les auteurs

***Damien SEGUY** dirige la BU Consulting du groupe Alter Way. Personnalité incontournable du Libre en France, il est l'un des fondateurs de l'Association Française des Utilisateurs de PHP (AFUP). Reconnu internationalement, il œuvre pour la promotion de la plateforme LAMP*

***Jean-Marc FONTAINE** est consultant senior chez AW Consulting et responsable de l'offre PHP chez Alter Way Solutions. C'est l'un des meilleurs experts français du Zend Framework.*

ALTER WAY Hosting

Alter Way Hosting (NEXEN) est leader de la conception, l'ingénierie, l'hébergement et l'infogérance des technologies webs Open Source.

*Avec une gamme complte de solutions et de services, Alter Way Hosting répond à tous les besoins d'hébergement que ce soit sur plate-forme mutualisée, virtualisée, dédiée ou architecture en 24*7 avec des engagements (SLAs) d'intervention (GTI) et de rétablissement(GTR).*

Des clients comme Generali, le Ministère de l'Agriculture et de la Pêche, Prisma Presse font confiance à Alter Way Hosting et bénéficent de solutions d'hébergement performantes et sécurisées via un réseau IP dédié et 3 datacenters.

Alter Way Hosting a su cultiver depuis plusieurs années une maîtrise optimale des technologies Open Source et un savoir-faire reconnu dans l'exploitation des sites internet, intranets et applications Open Source.

Alter Way Hosting souhaite ainsi offrir aux entreprises un environnement technologique et humain serein gage d'une collaboration équilibrée, performante et pérenne.

Gràce au partenariat avec ip-label.newtest, Alter Way Hosting permet à ses clients de certifier les engagements de qualité de services contractuels.

ALTER WAY Formation, la formation pour l'Open Source

Alter Way Formation (Anaska) est la référence formation dans le monde du libre. Ses formateurs hautement qualifiés travaillent également sur des projets concrets, professionnels et communautaires.

L'entité Formation collabore directement avec les éditeurs et les communautés liés aux technologies du libre. MySQL, SUN, Talend et Ingres lui font confiance pour leurs sessions.

ALTER WAY Consulting, l'expertise à code ouvert

ALTER WAY Consulting met à la disposition des entreprises et des administrations l'expertise de haut niveau en technologies Open Source qui caractérise le groupe Alter Way.

Les interventions de ses consultants visent à apporter une contribution décisive aux projets de changement, d'innovation, à éclairer les choix technologiques des clients ou à lever les freins qu'ils rencontrent dans leur utilisation de solutions Open Source.

ALTER WAY Solutions, le savoir-faire avancé

Alter Way Solutions rassemble des équipes reconnues pour leur expertise sur les principaux types de solutions open source.

1/ Les solutions applicatives :

- *Portails X-Net / Internet : intégration de produits tels Drupal, eZ Publish, WordPress, Typo3, Joomla, Zope, Plone, Django, Liferay*
- *Web 2.0 / Collaboratif : solutions s'appuyant soit sur des développements spécifiques ou sur l'intégration de produits type CMS, Moteurs de blogs, Wiki*
- *GED / ECM : intégration de produits open source tels Alfresco, Anakeen, Nuxeo*
- *E-Commerce avec Magento, Prestashop, OS Commerce*
- *Business Intelligence : Alter Way Solutions couvre tous les besoins du SI en terme d'applications décisionnelles, via le développement de solutions éprouvées :*
  - *ETL : Talend, Kettel, Octopuss*
  - *Bases de données multidimensionnelles, OLAP : Mondrian ou Palo*
  - *Stockage des données : MySQL, Ingres, PostgreSQL*
  - *Reporting :JasperReport Birt, JfreeReport*
  - *Gestion des PKIs : FreeMetrics*
  - *Datamining : Weka*
  - *CRM : intégration de SugarCRM, Open ERP, Open CRX*
  - *BPM : Intalio, Bonita, jBPM*
- *Développement spécifique : Alter Way Solutions propose une approche industrielle basée sur un cycle de développement sur-mesure s'appuyant sur des outils Libres. Et ce, quel que soit l'environnement technique cible (PHP, Python, Java, Ruby).*

2/ Les solutions d'infrastructure :

- *Supervision : Alter Way Solutions a développé une expertise pointue de la supervision en apportant à ses clients des réponses concrètes en matière d'intégration, de fonctionnalités et d'administration centralisée. L'approche Full GPL intègre le respect des standards universels garants de l'interopérabilité des systèmes. Parmi les technologies utilisées : Centreon, GLPI, OCS Inventory*

- *Serveur de messagerie : La solution d'Alter Way est basée sur Linux et Postfix, assurant ainsi l'indépendance logicielle. Elle s'intègre au SI de manière transparente autour des technologies d'unification des données Open Source (OpenLDAP) avec récupération des procédures d'exploitation. Elle peut intégrer un anti-virus et un anti-spam performants.*
- *Serveur d'infrastructure : L'accès à l'information pour les utilisateurs est sécurisé par une authentification sur un contrôleur principal de domaine, par une gestion des droits des groupes utilisateurs et utilisateurs individuels, apportant une maîtrise complète de l'accès à l'information. Les données sont également sauvegardées à fréquence déterminée par la mise en place d'une procédure de sauvegarde multi support. Le serveur de fichiers est un élément incontournable et nécessite d'être toujours disponible, c'est pourquoi Alter Way assure à ses clients la haute disponibilité de leur serveur.*
- *Filtrage et partage des connexions Internet : La solution proposée par le pôle Solutions du groupe respecte les politiques d'entreprise liées aux groupes utilisateurs, catégorisés et hiérarchisés par mode d'utilisation. L'administration de la solution permet notamment de filtrer par listes blanches, par listes noires regroupant plus de 15 thèmes dédiés. La solution est entièrement administrée via une interface web intuitive et personnalisable. L'intégration au sein du SI s'interconnecte avec la technologie LDAP de centralisation de l'information .*
- *Sécurité : Alter Way Solutions déploie ASTARO. Cette solution se présente sous la forme de serveurs prêts à emploi qu'il est possible d'administrer via une interface Web évoluée, permettant de s'adapter avec souplesse au SI. Le cœur du système d'exploitation est un noyau Linux 2.6 optimisé et éprouvé. Le système d'exploitation en lui-même est une base de Suse Linux Enterprise Server V10.*
- *IPAM : ECL IP'S est une solution ouverte dédiée à la gestion des plans d'adressage et de nommage IP et des services DNS et DHCP. S'enrichissant de nouvelles fonctions répondant aux demandes de ses utilisateurs et s'adaptant en permanence aux évolutions techniques de ses domaines cibles, elle a gardé sa philosophie d'origine : être un outil fiable et efficace d'administration de réseaux et de services IP.*

Table des matières

1 Introduction

Le constat est clair : PHP est définitivement implanté dans les DSI. Toutes les entreprises en utilisent dans leur DSI, parfois même sans le savoir. Dailymotion s'en sert pour son infrastructure de distribution de vidéo à la demande; Essilor, comme les premiers ministres de France, du Luxembourg et de Belgique, des associations comme Médecins sans Frontières l'ont adopté pour leurs portails institutionnels; Renault et le groupe HEC l'utilisent pour leurs CRM.

Au fond, la question ne se pose plus de savoir si PHP est crédible : il a dépassé le stade de l'expérimentation.

L'évolution d'une nouvelle technologie passe par trois phases distinctes lors de sa reconnaissance par l'entreprise :

- la phase de tests, où la technologie doit rassurer les utilisateurs quant à sa capacité à rendre des services productifs ;
- la phase de missions critiques, où des projets importants s'appuient sur la technologie ;
- la phase de mission stratégique, où la technologie est adoptée comme norme principale.

Lors de la première phase, les atouts maîtres de PHP étaient son caractère Open Source et pragmatique. Facile à installer, agile à utiliser avec diverses bases de données et technologies, PHP s'est infiltré dans l'entreprise via le marché de niche des applications de confort : toutes ces applications, à durée de vie très courte, mais tellement utiles au quotidien. Elles tombent hors du radar de la DSI, et sont souvent réclamées avec beaucoup d'impatience par les utilisateurs finaux. Pour gérer des notes de frais, un carnet d'adresses ou un planning commun, PHP sait composer avec toutes les technologies en place et fournir des services immédiats. De plus, avec des applications ciblées et bien reconnues, comme Mantis, SugarCRM, Magento, Drupal, il satisfaisait des clients internes aux calendriers plus exigeants.

Avec la deuxième phase, ce n'est plus la technologie qui fait la différence : ce sont les pratiques de développement qui sont maintenant au cœur de l'action. Les projets continuent de grossir en taille : plus de développeurs, d'utilisateurs et de durée de réalisation. Il apparaît alors une nouvelle gamme de défis à relever : savoir planifier le développement, dialoguer avec la communauté d'utilisateurs, standardiser la qualité du code. L'industrialisation est simplement la prise en compte de tous ces nouveaux acteurs dans le processus de développement.

Le défi porte surtout sur le volume de travail à consacrer aux tâches de gestion du projet. Certaines tâches faisaient partie du processus d'évolution du projet, mais doivent maintenant être gérées différemment. En début de projet, avec un seul utilisateur, un coup de téléphone permet de résoudre un bogue et de mettre à jour des spécifications. Lorsqu'il y a 10 utilisateurs, et autant de bugs différents, le temps à passer au téléphone va être radicalement augmenté. Et quand ce sont les clients qui appellent, le syndrome des interruptions permanentes va vite s'imposer.

Plus que jamais, il est important d'adopter des méthodologies de développement et des outils adaptés pour les grands projets. PHP n'est pas en peine à ce niveau. Son expérience de l'Open Source lui donne une maturité suffisante pour être utilisé par des équipes de grande taille. PHP adopte les pratiques standards de l'informatique, et s'adapte aux outils existants. Et au-delà même, il apporte sa propre communauté et un vif esprit de collaboration.

2 Maîtriser le cycle de vie d'un projet PHP

PHP a été créé le 8 juin 1995 pour répondre à des problématiques Web.

A cette époque, Perl occupait trop de ressources pour que le créateur, Rasmus Lerdorf, puisse l'utiliser sur une simple page de CV. Il a ainsi revu l'approche des applications Web et conçu une plate-forme de développement pragmatique et efficace.

Cette description de compteur de pages de CV est restée collée à PHP. Publié à code ouvert, il est pourtant entré rapidement dans une phase d'expérimentation auprès des internautes et des entreprises. Il s'agissait pour cette nouvelle technologie de passer sous les fourches caudines de l'utilité : développements rapides, montée en charge sur le grandes architectures, capacité à évoluer et interfaces avec toutes les technologiques utiles pour publier des informations sur le Web et s'interfacer avec le SI. Durant cette première partie de sa vie, PHP a conquis le coeur d'une communauté toujours plus grande, et est entré dans la vie de nombreuses entreprises.

Depuis, PHP est poursuivi par certaines de ses pratiques originelles. Plate-forme parfaite pour les prototypes, nombre d'entre eux se sont retrouvés en production et ont évolué en de grands et vénérables sites. Dans l'ensemble, l'approche de développement reste encore trop souvent réactive et peu pro-active. Les projets adoptent souvent des capacités de réactions rapides : dès qu'un problème est détecté, il est remonté aux développeurs, analysé, corrigé et le code source est publié à nouveau.

"Fail early, fail often" a même été une devise pour un temps : PHP était parfait pour tester une idée rapidement et de manière économique. L'important était alors de détecter la fausse route le plus tôt possible (Fail Early), pour ne pas perdre trop de ressources ou de temps. La philosophie des sites Web en PHP est loin des techniques de lancement de fusée spatiale.

Pour les projets d'envergure, la pratique montre différents signes de vieillissement classiques des logiciels : les premières itérations sont rapides et spectaculaires; progressivement, avec chaque nouvelle version, le temps de développement s'allonge et les versions s'espacent. Le code mis en production, testé manuellement, rencontre des problèmes inattendus en production. Il faut donc consacrer de précieux moments à corriger des bogues qui, détectés trop tard, sont arrivés jusqu'en production. Ou bien il faut se plonger dans du code historique, avec plusieurs couches de corrections et de mises à niveau depuis PHP 4, PHP 5, l'ère des frameworks maison ou des frameworks objets. Tout cela s'accumule comme la poussière dans une maison trop longtemps fermée. Finalement, on n'ose plus entrer dans ce code et, le cas échéant, on fait appel à un audit pour pouvoir en reprendre la maîtrise.

PHP a toujours eu une approche pragmatique et efficace, qui lui a permis de compenser ces travers. Les méthodologies agiles sont déjà adoptées par de nombreuses équipes Web. Les IDE et les frameworks portent l'essentiel de l'effort de l'industrialisation. Ils assurent la promotion des tests unitaires, des composants réutilisables et des méthodologies. PHP n'a ni l'apanage, ni la primeur de ces approches, qui sont parfois aussi vieilles que l'informatique elle-même : il a toujours pratiqué l'emprunt du meilleur auprès des autres technologies.

Parmi les pratiques répandues dans l'industrie, on trouve certaines approches récurrentes. Nous avons pensé qu'il était bon de les rappeler aujourd'hui, car elles sont déjà matures pour PHP. Malheureusement, elles ne sont que très rarement rassemblées toutes dans un même projet.

- Faire appel à un audit expert
- Former les équipes
- Appliquer une convention de programmation
- Utiliser un dépôt de code
- Utiliser un framework
- Adopter un IDE de développement.

Par ailleurs, l'innovation est très présente en PHP et de nouvelles pratiques s'adaptent au Web et à PHP. Elles viennent compléter les approches traditionnelles. Elles sont novatrices et offrent de beaux défis, et des champs d'exploration encore vierges.

- Tests unitaires, IHM et fonctionnels
- Déploiement automatique

- Analyse statique
- Discipline et méthode de programmation
- Surveillance continue de la qualité
- Collaboration avec les utilisateurs.

Voyons maintenant toutes ces approches en détails.

3 Pratiques actuelles

3.1 Faire faire un audit par un expert

A un moment de la vie d'un projet, on a besoin d'un diagnostic et donc, d'un expert. Le projet présente des symptômes naissants de gangrène (sécurité, performance, maîtrise du code, sentiment de maîtrise, etc.). Les audits sont particulièrement intéressants lors de la reprise d'une activité, ou en cas de disparition complète de l'équipe de développement : reprendre en main un existant se fait bien après une évaluation externe ou interne et il est souvent plus efficace de passer par un service externe.

Pour cela, il faut faire appel à un expert : ce dernier vient réaliser un état des lieux de l'application et de son organisation. Les missions portent souvent le nom d'un des symptôme principaux :

- Audit de sécurité
- Audit de performances
- Audit de qualité logicielle
- Audit de l'application

Cette dernière dénomination est générale. Dans la pratique, elle recouvre les trois problèmes précédents et signale des problèmes graves. C'est un manque de confiance dans l'application et un manque de direction dans son développement.

Les sociétés de services capables de mener un audit PHP sont actuellement peu nombreuses. Elles doivent disposer d'experts, de préférence reconnus par la communauté et d'une vaste expérience de développement et de menée de projets. De bonnes références sont aussi un atout. La pratique des audits en général, même si elle n'est pas appliquée à PHP, est alors un atout : savoir présenter un rapport est un métier en soi.

Un audit d'une plate-forme Web prend généralement entre une et trois semaines, et impose une mise à disposition d'informations assez importantes. Le code source, les configurations serveurs, l'architecture de la plate-forme, les acteurs du projets, la base de données et sa volumétrie. Dans tous les cas, il faut exiger un engagement à confidentialité de la part des experts.

Les audits doivent toujours intégrer l'utilisation des outils automatiques. Ces outils sont faciles à utiliser, et couvrent beaucoup de surface. Ils sont limités dans la profondeur de leur attaque, mais permettent d'identifier immédiatement des problèmes simples et spectaculaires ou de mettre en avant des problèmes potentiels qui seront analysés par un expert. Ils permettent aussi de vérifier les vulnérabilités qui ont été publiées : ce sont les problèmes les plus fréquents.

Il existe plusieurs outils d'audit de code PHP.

Outil	**Objectif**
RATS	Analyse le code PHP pour identifier des failles courantes
Pixy	Analyse l'utilisation des variables PHP pour détecter les injections SQL et XSS
CodeSecure	Intégré dans les IDE, analyse la propagation des variables corrompues dans le code

Les outils automatiques d'analyse de code sont immatures : ils apportent des retours superficiels, quoique souvent spectaculaires. Ils gagneront à être utilisés dans le cadre de la maîtrise du code tout au long du développement.

Les audits d'applications PHP sont aujourd'hui peu sollicités. La taille des projets permet souvent d'abandonner la base de code en cours et de reprendre à partir de zéro. C'est une règle bien connue des équipes de développement, qui choisissent souvent ce type de solution. Entre l'audit et la réécriture totale, il faut continuer la promotion des bonnes pratiques et de leur application.

Enfin, le recours à un audit PHP doit toujours être considéré comme un dernier recours : c'est une étape qui va coûter très cher. Des méthodes existent pour prendre en compte les problèmes de qualité de code et de sécurité plus en amont, durant le développement ou bien dès la conception et l'architecture.

3.2 Formation des équipes

PHP est reconnu pour sa simplicité d'utilisation et de prise en main. Nombreux sont les autodidactes qui décident de faire leurs premiers pas sur le Web avec PHP et évoluent vers de nouvelles responsabilités. Cette approche tire partie de l'expérience acquise sur le terrain et fournit des résultats rapides et tangibles.

Les équipes de développement sont fréquemment organisées autour d'un gourou : souvent en interne, parfois en externe, le gourou a pour vocation de guider les développeurs en devenir entre les fonctionnalités et les méthodes que propose PHP pour résoudre les problèmes. Un gourou est précieux, mais il est difficile d'en avoir un de disponible et ils sont rares. Au final, la formation est la meilleure solution pour élever le niveau de jeu de son équipe.

Pour maintenir un bon niveau de compétences techniques et méthodologiques dans une équipe, la formation régulière est un atout indispensable. Selon le temps disponible et le budget alloué à la formation, plusieurs options se présentent.

Accessible à tout un chacun, une veille technologique régulière est essentielle. Grâce aux techniques de syndication (RSS, Atom, etc.) et aux agrégateurs, ces outils qui centralisent l'accès aux flux d'information, mettre en place une veille PHP est relativement simple. En fait, obtenir une masse d'information sur l'actualité PHP et connexe est aisée, mais le ratio signal/bruit risque d'être faible.

Il y a plusieurs possibilités pour diminuer la part du bruit dans une veille :

- Utiliser les filtres des agrégateurs afin de n'extraire que les articles intéressants de flux de syndication trop généralistes ;
- S'abonner à des sites qui effectuent une sélection des articles les plus pertinents dans un domaine ;
- Identifier des personnes ayant les mêmes centres d'intérêt et partageant le fruit de leur veille sur des services de bookmarks sociaux (Delicious, Diigo, etc.).

Une activité sérieuse de veille est nécessaire dans une équipe importante : il est intéressant d'attribuer ce rôle à une personne curieuse, qui sera chargée de partager le fruit de sa veille avec le reste de l'équipe. Cette approche est formatrice. Elle permet aussi de dégager des intérêts techniques ou des discussions sur l'actualité PHP et des technologies, dont les développeurs sont friands.

Pour avoir plus de recul sur les évolutions de fond de PHP, la formation professionnelle reste le meilleur outil.

Les organismes de formation proposent des sessions concentrées en moins d'une semaine. Le formateur est spécialement entraîné pour faire la synthèse des évolutions de fond de PHP, et les faire comprendre correctement aux stagiaires. La formation comporte généralement deux phases qui alternent :

- Des présentations magistrales détaillant les aspects théoriques des technologies;
- Des travaux pratiques où l'on applique les notions précédentes à des cas concrets.

Les formations peuvent être dispensées au sein d'une entreprise uniquement pour le personnel de celle-ci, on parle alors de formation intra-entreprise, ou bien dans un organisme de formation pour des personnes venant de sociétés différentes, il s'agit dans ce cas de formation inter-entreprises.

Les formations sont un pied à l'étrier pour aborder une nouvelle technologie ou un nouvel outil. Il ne faut pas penser que des développeurs partant de zéro seront parfaitement opérationnels dès la fin de la formation. En revanche, les quelques jours qu'ils y auront consacrés doivent être vus comme un investissement pour éviter une auto-formation nécessairement plus longue et comportant le risque de faire l'impasse sur certains éléments importants ou de partir sur de mauvais usages.

Parmi les très nombreux sources d'informations disponibles, voici les plus notables :

Nom	**URL**	**Langue**
Planet PHP France	http://www.planete-php.fr/	Français
Planet PHP	http://www.planet-php.net/	Anglais
PHP Developper	http://www.phpdeveloper.org/	Anglais
PHP sur Google News	http://news.google.com/news?pz=1&ned=fr&hl=fr&q=PHP&scoring=n&output=rss	Francais ou anglais
Delicious	http://delicious.com/tag/PHP	Anglais

La formation des équipes se concentre sur PHP et sa technique. Les notions présentées y sont principalement techniques et s'attachent à

garder le développeur au top niveau de ce que la plate-forme lui propose. Il est indispensable de garder ses équipes à l'heure concernant les problématiques PHP, ou encore sur des aspects critiques comme les performances, la sécurité ou les outils d'industrialisation.

3.3 Employer une convention de programmation

De la même manière que chacun d'entre nous a une écriture qui lui est propre, chaque développeur a ses habitudes de travail. Le formatage du code et le nommage des éléments varient beaucoup d'un développeur à un autre. Pourtant, le commun des mortels utilise une langue écrite, le français dans ce texte, qui comporte des règles et des conventions. Le respect de ces règles tend à supporter le message que veut faire passer l'auteur et permet souvent d'en vérifier le contenu. Les conventions de code jouent le même rôle vis-à-vis des applications.

C'est d'autant plus important que les équipes de développement ont tendance à croître en taille. Entrer dans le code d'un programmeur qui utilise d'autres conventions que les notres, ce qui est souvent le cas, est une barrière à la compréhension. D'ailleurs, une grosse partie du temps est passée à reformater ce code, simplement pour le mettre dans un format lisible et agréable. Au lieu de laisser libre cours à des conventions parfois contradictoires, il est recommandé d'en choisir une seule, commune à toute l'équipe.

Code mal écrit :

```
<?php
function fibonacci($n)                                                    {
$i = $a = $b = 0;
while($i < 0)                    {$c = $a + $b; $a = $b; $b = $c; $i++}
return $b;
}
?>
```

Code aux conventions :

```
<?php
function fibonacci($n) {
   $i = 0;
   $a = 0;
   $b = 0;
   while($i < 0) {
     $c = $a + $b;
     $a = $b;
     $b = $c;
     $i++
   }
   return $b;
} 
?>
```

Certains langages imposent des conventions de programmation, comme Python. Ce n'est pas le cas de PHP. Il n'y a pas, à ce jour, de convention officielle.

Il existe toutefois plusieurs conventions de programmation découlant souvent des communautés ou de certains projets d'envergure. Celles-ci expliquent plus ou moins en détails comment le code source doit être formaté (indentation, encodage, placement des signes, etc.) et comment les éléments qui le constituent doivent être nommés (fichiers, classes, méthodes, variables, etc.). Chacun est donc libre d'utiliser celle qu'il souhaite. Il existe trois stratégies pour choisir une convention de programmation :

- Adopter une convention existante ;
- Adapter une convention existante ;
- Créer sa convention.

Créer sa propre convention est souvent le choix qui paraît le plus naturel. On est ainsi sûr d'avoir une convention qui réponde parfaitement à ses besoins et représente le plus petit effort d'adaptation de la part de l'équipe en place. Par contre, cet effort de standardisation fait partie du contrat social de la convention : l'important est de sensibiliser les développeurs au fait qu'un autre développeur puisse se sentir aussi à l'aise dans son code.

Il est alors naturel de partir d'une convention existante et de la modifier ou de la compléter selon ses besoins spécifiques. Cela demande un travail réduit mais là encore l'écart avec une convention standard devra être intégré par les nouveaux arrivants ou les prestataires travaillant pour la société.

La solution la plus courante est d'adopter une convention existante et de s'y tenir. D'ailleurs, les frameworks proposent toujours une convention de programmation, qu'il est pratique d'adopter.

Le choix d'une convention est généralement moins important que la discipline qui l'accompagne. Comme souvent, cela tourne aux querelles de clocher et il est difficile de mettre tout le monde d'accord. Une fois un choix adopté, il est primordial de respecter la convention de manière systématique, sous peine de perdre progressivement la maîtrise de la présentation. D'ailleurs, les points d'un logiciel où le nombre de violations de convention de code est le plus grand représente toujours une raison pour un audit.

Il existe plusieurs conventions de programmation dans le monde PHP. Trois d'entre elles sont le plus souvent citées :

- **PEAR** : La célèbre bibliothèque de composants PHP utilise une convention bien conçue et documentée qui a été reprise par de nombreux projets Open Source ;
- **Zend Framework** : Pour son framework, la société Zend s'est basée sur la convention de PEAR et l'a complétée ;
- **PHP Coding Standard** : Cette convention est maintenant ancienne mais elle est bien conçue.

Le choix final d'une convention est souvent une question d'habitudes et de goûts personnels cependant le choix peut être induit par l'usage d'un outil comme un framework ou un CMS. Ceux-ci ont généralement une convention de codage qu'il est judicieux d'utiliser afin de s'assurer de l'homogénéité de l'application développée.

Vérifier l'application d'une convention de programmation peut rapidement devenir une tâche titanesque. Heureusement, des outils ont été créés afin de simplifier l'essentiel voire la totalité des vérifications :

- **PHP_CodeSniffer** : Ce paquet PEAR est l'outil de référence pour vérifier l'application d'une convention de programmation. Il propose de valider plusieurs des conventions les plus connues dont celles de PEAR et de Zend Framework mais également de créer des vérifications supplémentaires. Par ailleurs, il permet également de vérifier le code Javascript et CSS ;
- **PHPCheckStyle** : Bien que moins connu et plus limité que PHP_CodeSniffer, cet outil peut rendre des services ;
- **PHPCodeBeautifier** : Il se charge de mettre votre code à votre convention, sur Windows.

Choisir une convention de programmation n'est pas un acte anodin car le code source d'une application sera manipulé tout au long de sa vie. Il est recommandé de le choisir avec l'équipe, pour une meilleure adoption, et pour identifier les conflits que cela peut engendrer.

Le plus important reste de la respecter : des outils automatiques d'analyse permettent de tester régulièrement le code et de produire des indicateurs sur la discipline. Tout comme la sécurité, cela se travaille au quotidien.

3.4 Utiliser un dépôt de code

Le développement d'une application est rarement aussi linéaire que les planifications de projets le laissent entendre. Il y a des essais, des erreurs, des tâtonnements. Quand on travaille à plusieurs, il est encore plus compliqué de s'organiser pour publier un code propre sans se marcher sur les pieds : le premier symptôme du besoin d'un dépôt de code survient lorsque deux développeurs travaillent sur le même fichier et que le dernier écrase le code correctif du premier.

Les petites équipes choisissent souvent de centraliser leur code à l'aide d'un partage réseau sauvegardé régulièrement dans des archives. Cette solution est simple à mettre en place, mais la pratique montre vite ses limites :

- Il n'y a pas de gestion de conflits, chacun peut écraser le travail de ses collègues par mégarde ;
- Il est impossible d'avoir un historique des modifications apportées au code ;
- Il est très difficile de gérer des expérimentations.

Il est alors temps de passer à un dépôt de code, qui sait gérer les versions et les conflits.
Les dépôts de code ont été créés spécialement pour régler ces problèmes, qui sont communs à toutes les technologies qui gèrent des codes sources. Ces logiciels stockent un historique de chacune des modifications apportées : on appelle ces modifications des révisions. Grâce à cet historique, il devient possible de :

- Savoir qui a écrit, quel code et quand ;
- Visualiser les modifications apportées entre deux révisions ;
- Revenir à une version précédente du code si nécessaire.

Les versions sont entretenues sur un serveur appelé un dépôt. Un dépôt est un système client-serveur. Le serveur stocke l'ensemble du code source. Il est chargé de conserver l'historique des modifications, de veiller au respect des droits d'accès, d'incrémenter les numéros de versions, de gérer les tags de versions, de signaler les conflits, etc.

Les logiciels clients se connectent au serveur afin de récupérer le code source sur lequel travailler. Une fois ce travail effectué, les modifications apportées à ce code source sont renvoyées sur le serveur. Les autres clients peuvent ultérieurement récupérer ce code source mis à jour et continuer leur travail sur leurs postes respectifs.

La terminologie appliquée aux versions est empruntée aux arbres. Ainsi, on parle de tronc pour désigner l'endroit où est stocké le code principal et de branches pour les variations du tronc. Il existe plusieurs stratégies pour gérer ces branches (branches expérimentales, branches de maintenance, branches par développeur ou par équipe, etc.)

En complément du tronc et des branches, il existe des tags qui sont des «photos» du code à un instant donné. C'est à l'aide de tags que l'on gère les versions des applications, sans empêcher son développement.

Il existe de nombreux outils tant Open Source que propriétaires pour implémenter un dépôts de code.

Dans ce domaine, le serveur Open Source CVS a longtemps été l'outil le plus populaire. Il est aujourd'hui remplacé par un nouveau serveur appelé Subversion, souvent abrévié SVN, qui reprend les mêmes bases que CVS, et corrige nombre de limitations que la pratique a mis à jour. Subversion est actuellement le dépôt de code centralisé de référence. Puissant et fiable, il est utilisé par de très nombreuses sociétés et projets Open Source partout dans le monde : PHP lui-même a fait la migration de CVS vers SVN.

Actuellement, les gestionnaires de dépôts dits « décentralisés », par opposition aux gestionnaires traditionnels qui sont dits « centralisés », ont le vent en poupe. Leur fonctionnement local est le même, à la différence majeure qu'il n'y a pas de serveur central. Chaque client est également serveur et à ce titre s'organise avec les autres clients pour entretenir un code source unifié. La souplesse ainsi gagnée se fait au prix d'une plus grande intervention humaine dans le processus. Git est probablement l'outil le plus emblématique de ces gestionnaires. Il a été créé par Linus Torvalds, également créateur de Linux, et est utilisé pour gérer le code du noyau de ce dernier.

Nom	Editeur	Type
Subversion	Open Source	Centralisé
CVS	Open Source	Centralisé
Git	Open Source	Décentralisé
Mercurial	Open Source	Décentralisé
Bazaar	Open Source	Décentralisé
Rational ClearCase	IBM	Centralisé
Perforce	Perforce Software	Centralisé
Visual SourceSafe	Microsoft	Centralisé

Mettre en place un dépôt de code est une étape cruciale dans un processus d'industrialisation de ses développements. On y vient pour des raisons de gestion quotidienne du code source, mais le VCS (Version Control System) va vite devenir la clé de voute de l'industrialisation : le code de l'entreprise est le code qui est dans le dépôt et nombre d'outils secondaires viennent y prendre le code : l'usine de code, les outils de tests, les administrateurs qui déploient le code.

3.5 Utiliser un framework

PHP présente une richesse fonctionnelle incroyable : plus de 7000 fonctions documentées, des centaines d'extensions. Il s'adapte facilement aux diverses problématiques qu'on rencontre lors du développement d'applications Web. Il propose souvent plusieurs solutions pour le même problème. Et cette richesse a cependant un revers : elle peut rapidement se transformer en chaos si l'on n'y prête pas attention.

Initialement, PHP a été créé pour pouvoir mélanger au sein d'un même script du code de traitement, l'accès aux données et l'affichage HTML. Le code était confus, la structure peu claire et la maintenance compliquée. Aujourd'hui, la séparation du contrôleur, du template (ou gabarit) et du modèle, est une bonne pratique courante.

Et pour aller plus loin dans la structuration du code, les frameworks sont alors un passage obligé. Ils sont d'ailleurs nombreux et très dynamiques.

Les frameworks ont évolué pour répondre à différentes problématiques. Un framework fournit un ensemble de composants, d'outils et de conventions permettant le développement d'applications.

- Les composants sont livrés sous forme de bibliothèques et font gagner du temps en évitant de réinventer la roue. Les frameworks sont d'ailleurs généralement capables d'adopter des composants issus d'autres frameworks ;
- Les outils permettent de gagner du temps de mise en place, en créant rapidement des squelettes à compléter, ou en assurant la création directe de tests unitaires ;
- Les méthodes sont un ensemble de convention et d'organisations qui permettent d'assurer un bon niveau de sécurité et de maintenance, ainsi que l'homogénéité du code.

Les frameworks utilisent généralement la Programmation Orientée Objet de manière intensive, car ils exploitent les capacités de généricité et de réutilisabilité du modèle objet. Ce n'est toutefois pas une obligation.

PHP compte de très nombreux frameworks : plus de 400 à ce jour. Ils adoptent des philosophies différentes, parfois contradictoires, et parfois astucieusement réconciliées : MVC, strut, événementiel, Full stack, faible couplage, etc. Seule une poignée d'entre eux est en mesure de répondre aux besoins des entreprises pour développer des applications importantes.

Parmi eux, les deux plus connus sont Symfony et Zend Framework. Leurs modes de développement sont similaires. Ces deux frameworks sont développés et supportés par une société, respectivement Sensio Labs et Zend Technologies. Les deux frameworks bénéficient d'une communauté importante de développeurs bénévoles et professionnels.

Au niveau conceptuel, Symfony et Zend Framework sont basés sur le motif de conception MVC : Modèle-Vue-Contrôleur. Il s'agit d'un concept d'architecture logicielle qui prône la séparation stricte entre l'accès aux données, leur traitement et leur affichage. Au niveau technique en revanche les approches divergent. Symfony se positionne comme un framework intégré, qui encadre fortement le développement afin de permettre un développement rapide d'applications. Le Zend Framework se présente plus comme une bibliothèque de composants fortement découplés dans laquelle le développeur va pouvoir piocher.
Il faut noter que le développement de chacun de ces frameworks va dans le sens d'une rencontre entre les deux approches. Ainsi, Symfony découple de plus en plus ses composants tandis que Zend Framework propose de plus en plus de composants et d'outils de haut niveau.

Les frameworks aident véritablement les équipes de développement à structurer les projets. C'est particulièrement important pour les projets

d'envergure. Le choix du framework est important, car il a des impacts forts sur l'organisation de l'équipe de développement, ou son acceptation.

Le choix du framework prend en compte différents aspects : sa maturité, la taille de sa communauté, l'offre de services autour du framework. Cet univers est très concurrentiel, car le web est en innovation permanente. Les codes vieillissent vite et s'encombrent parfois trop vite de fonctionnalités qui sont abandonnées.

Leur ouverture permet aussi d'envisager une utilisation des frameworks avant-gardiste : contribuer à un framework permet de mieux le comprendre et de l'intéresser à des sujets qu'il n'a pas toujours envisagé de traiter. Avec, à la clé, une diffusion très importante dans toute la communauté.

3.6 Adopter un IDE de développement

Les scripts PHP sont de simples fichiers de texte. Pour les créer un éditeur de base comme le BlocNotes de Windows suffit. Certains s'en contentent d'ailleurs, ou bien utilisent des outils légèrement plus avancés, comme Notepad++, qui apporte les premières fonctionnalités adaptées. On parle alors d'éditeurs de texte.

Mais pour être efficace, il faut bien plus que les fonctionnalités de base proposées par ces outils. Si un pilote de ligne peut manœuvrer un avion en commandes manuelles, la tâche lui sera grandement facilitée par l'usage des systèmes modernes d'assistance. Il en va de même pour l'édition de PHP.

L'aide à la saisie, par exemple, permet de détecter des erreurs de syntaxe ou des coquilles dans les noms de variables très tôt et évite de passer par un processus d'erreur / débogage pour les corriger : autant de temps de gagné.

Pour répondre à des besoins de plus en plus avancés, les éditeurs de texte ont fait place à des IDE (Integrated Development Environment). Un IDE est un programme qui regroupe un ensemble d'outils pour le développement d'applications, avec, au minimum, un éditeur de texte et un débogueur.

Les IDE PHP proposent généralement les fonctionnalités suivantes :

- Colorisation syntaxique ;
- Autocomplétion ;
- Intégration aux dépôts de code ;
- Intégration avec un ou plusieurs frameworks ;
- Débogueur ;
- Profileur ;
- Intégration d'outils externes (Tests unitaires, déploiement, gestion de base de données, éditeur UML, etc.).

Dans le monde PHP, le débogage d'une application reste trop souvent artisanal. Les valeurs des variables sont affichées avec des fonctions placées judicieusement dans le code et une méthode dichotomique. Cette technique est fastidieuse et souvent peu efficace. Elle est largement réduite par l'utilisation de débogueurs. Ces derniers sont couramment utilisés dans d'autres langages comme Java ou C++, où la phase de compilation empêche la modification du code à des fins de tests. La pression pour l'utilisation de ces outils est donc beaucoup plus forte que dans le monde PHP.

Le débogueur permet d'exécuter pas à pas une application afin de s'assurer que le cheminement à travers le code est bien celui attendu. On peut à tout moment consulter le contenu de chaque variable. Il est possible de définir des points d'arrêt. Ce sont des endroits dans le code où le débogueur stoppera momentanément l'exécution de l'application afin de ne pas avoir à parcourir pas à pas l'ensemble du déroulement de l'application à partir de ce point, ou consulter le contenu des variables.

Parallèlement au déboguage dont la vocation est d'aider à diagnostiquer les bogues, il existe le profilage dont le but est de déterminer les endroits qui ralentissent l'exécution de l'application. Un profileur s'interface avec le coeur de PHP afin de recueillir des informations sur le comportement de chacun des éléments du code au cours de l'exécution.

Ainsi, on peut visualiser, généralement sous forme d'arbre, les éléments qui interviennent lors de l'exécution de l'application. Pour chacun, il est possible de savoir combien de fois il a été appelé, combien de temps il a fallu pour qu'il fasse son travail et éventuellement combien de mémoire il a utilisé. C'est extrêmement utile car l'expérience montre que les goulets d'étranglement ne sont pas toujours là où l'on pense.

Au delà des outils fournis en standard, les IDE sont souvent modulaires. Il est donc aisé d'en étendre les fonctionnalités à l'aide d'extensions libres ou commerciales.

Il est recommandé de définir un IDE de référence qui doit être utilisé pour tous les développements au sein de la société. Cela permet de standardiser les postes, de s'assurer que chacun peut travailler efficacement avec des outils conformes aux besoins et aux contraintes.

L'offre d'IDE PHP est très vaste et variée tant du côté des outils libres que commerciaux.

Nom	Editeur	Commentaire
Aptana	Aptana Inc.	Initialement orienté HTML/CSS, cet éditeur propose dorénavant une gestion avancée de PHP
NetBeans	Sun Microsystems	Récemment arrivé dans le monde PHP, cet éditeur propose néanmoins des fonctionnalités de haut niveau
PDT	Open Source	Extension pour Eclipse développée par Zend Technologies et la communauté
Zend Studio	Zend Technologies	Basé sur PDT, cet éditeur apporte des fonctionnalités avancées et s'interface finement avec Zend Framework

Au delà de ses fonctionnalités propres, un IDE est conçu pour s'intégrer au sein d'une usine de développement. Certains éditeurs comme Zend Technologies proposent des produits dont Zend Server et Zend Plateform qui s'interfacent avec Zend Studio et permettent d'optimiser des serveurs de production, de déboguer à distance et même de rejouer des incidents a posteriori.

4 Outils et méthodes avancées

4.1 Tests d'application Web

L'erreur est humaine. Même les meilleurs développeurs écrivent parfois du code avec des bogues. C'est inévitable. Ces problèmes peuvent être techniques, le résultat d'un calcul est erroné, ou ergonomiques, un formulaire de l'application ne fonctionne pas comme attendu.

Tout développeur vérifie que le code qu'il a produit fonctionne. Cependant, plus l'application se complexifie et plus il lui devient difficile de vérifier toutes les combinaisons. Cela prend beaucoup trop de temps et les risques d'oublis sont importants. Les tests sont alors reportés à la phase de recette.

Cette phase permet à des testeurs qui peuvent être des personnes payées pour cette tâche ou bien directement les utilisateurs finaux de s'assurer de la conformité de l'application livrée avec les spécifications. Elle est longue et demande beaucoup de rigueur pour s'assurer que l'ensemble des scénarios utilisateurs ont été vérifiés. Un scénario utilisateur est un ensemble d'actions qui doivent aboutir à un résultat reproductible. Par exemple, mettre un produit dans le panier d'achat ou bien encore effectuer une recherche sur un produit.

Si des problèmes sont trouvés pendant cette phase de recette, les développeurs sont chargés de les corriger avant qu'une nouvelle phase de recette ne valide les correctifs. Ce cycle peut être long et donc coûteux. Par ailleurs, plus un bogue est constaté tardivement plus sa correction est difficile et donc coûteuse car un développeur est beaucoup plus apte à corriger rapidement un morceau de code une heure après l'avoir écrit que plusieurs semaines après. D'autant, que plusieurs semaines plus tard, il est probable qu'un autre développeur se voit assigner la correction du problème. C'est alors que les conventions de programmation l'aideront à s'imprégner plus rapidement du code.

Dans une application, aucune modification n'est triviale. Les effets de bord, c'est à dire les effets inattendus d'une modification a priori simple et maîtrisée, peuvent être très difficiles à repérer.

Comme pour le déploiement, l'automatisation des tests est la clé de la qualité d'un code. Nous allons plus particulièrement nous pencher sur trois types de tests : les tests unitaires, les tests IHM (Interface Homme/Machine) et les tests fonctionnels.

Tester unitairement le code d'une application consiste à isoler une unité de ce code afin d'en vérifier le comportement dans différents contextes. Par exemple, on testera individuellement chaque méthode d'une classe afin de s'assurer qu'en lui passant des paramètres déterminés, elle produit le résultat attendu. Les tests unitaires sont souvent accompagnés d'une vérification de la couverture de code, qui consiste à s'assurer que le test conduit à exécuter l'ensemble (ou une fraction déterminée) des instructions présentes dans le code à tester.

Les tests fonctionnels décrivent en langage non technique une fonctionnalité. Par exemple, pour vérifier le calcul d'un ratio comptable, on fournira les valeurs d'entrée et le résultat attendu.

Les tests IHM, ou tests de recette, quant à eux permettent de s'assurer que les comportements de l'application sont ceux attendus. Par exemple, lorsqu'on soumet ces valeurs à ce formulaire les messages d'erreur sont conformes aux attentes.

Lors de la création des tests, une erreur courante consiste à ne rédiger que les tests pour les actions qui doivent réussir. Il est tout aussi important de s'assurer que ce qui doit échouer échoue bien. Ainsi, la saisie

d'une valeur erronée dans un champ de formulaire doit être interceptée et le message correspondant affiché. Si la valeur est accepté sans erreur alors il y a un problème.

Une fois créés, ces tests peuvent être lancés de manière automatisée. Cela est beaucoup plus rapide car un analyseur logiciel effectue les vérifications bien plus efficacement qu'un humain et sans risque de distraction ou de fatigue. Ils peuvent également être lancés des dizaines de fois par jour sans lassitude ni problème d'horaires.

Une application reste rarement figée au cours de sa vie. Elle évolue régulièrement au gré des besoins de ses utilisateurs. Grâce aux tests automatisés, ce changement n'est plus vécu comme un stress. Les équipes sont plus sereines car sûres de la qualité du code de l'application.

Il existe deux types de modifications :

- la réécriture qui consiste à modifier le comportement d'une partie de l'application. Les tests doivent être mis à jour dans ce cas ;
- la refactorisation qui consiste à réorganiser et optimiser le fonctionnement d'une application sans en changer le comportement. Les tests doivent continuer de passer. Ainsi, si les tests passent toujours, le développeur est assuré de ne pas avoir cassé l'application en la modifiant.

Dans le monde PHP, il existe deux grands frameworks de tests unitaires : SimpleTest et PHPUnit. Conceptuellement proches, ces deux frameworks se distinguent par certaines fonctionnalités spécifiques comme un navigateur virtuel pour SimpleTest ou une extension permettant de tester facilement le code lié à une base de données pour PHPUnit.

Nom	**Editeur**	**Tests unitaires**	**Tests IHM**
SimpleTest	Open Source	Oui	Oui
PHPUnit	Open Source	Oui	Oui *(avec Selenium)*

Il n'existe pas à l'heure actuelle d'outils spécifiques à PHP pour les tests fonctionnels. Cependant, les tests fonctionnels étant peu liés aux aspects techniques, il est parfaitement possible d'utiliser des outils venus d'autres univers.

Nom	Editeur
Fitnesse	Open Source
GreenPepper	Open Source

Les tests, quelque soit leur nature, concourent à la qualité de code et à la tranquillité d'esprit des équipes.

Il est important de s'assurer de la qualité du code d'une application car sinon viendra un jour où il sera impossible de faire les modifications attendues et il faudra entièrement réécrire l'application pour repartir sur des bases saines. Cette pratique est au cœur des méthodes agiles de développement qui en font un élément crucial de leurs préceptes.

4.2 Intégration continue

La pratique des tests ajoute une étape dans le processus de développement : il faut faire tourner les tests unitaires avant de progresser, soit vers d'autres versions, soit vers le déploiement. A chaque fois, il faut mettre en place le système de tests, les faire tourner et réagir en fonction des retours d'erreurs.

Or, l'historique des résultats de tests unitaires est une source d'information intéressante : quels types d'erreurs reviennent le plus souvent ? Quelles fonctionnalités sont les plus sensibles à des modifications ? Quand se produisent les commits les plus erronés ? Comment le projet est-il surveillé durant l'absence du chef de projet ?

Une réponse à ces questions est fournie par l'eXtrême Programming : elle tend à fluidifier au maximum la transition depuis le développement jusqu'au déploiement. Pour cela, elle automatise le déploiement et l'exécution des tests unitaires.

Un serveur d'intégration continue est mis en place, sous la forme d'un outil fonctionnant en permanence. Régulièrement et indépendamment de l'effort de programmation, il va extraire le code PHP du dépôt de code, le déployer sur un serveur de test, puis collecter les résultats et les informations obtenues à chaque étape. Le tout est publié sous forme de graphiques d'évolution.

Par exemple, avec phpUnderControl, les indicateurs suivants sont rapportés :

- Métriques sur le code PHP (complexité, couplage, etc.) ;
- Vérification des conventions de programmation ;
- Nombre de tests unitaires présents ;

- Couverture de code ;
- Résultats des tests ;
- Génération de la documentation PHPDoc.

Il devient alors possible de suivre des indicateurs de qualité de l'ensemble de l'application. Cela sert à nombre d'intervenants :

- Le chef de projet, qui a un rapport régulier de l'état du projet ;
- Les membres de l'équipe, qui suivent le travail du groupe. Ils ont aussi une image immédiate de leur impact sur le projet : un commit de code non compilable se voit immédiatement dans les statistiques du projet ;
- Les clients et utilisateurs, qui ont plus de détails sur les rouages internes du projet. Cela donne des descriptions chiffrées sur le fonctionnement interne, et non plus des évaluations qualitatives.

Dans la pratique, l'intégration continue tend à réduire le temps de publication des applications, à réduire les problèmes d'intégration, et généralement à donner plus de confiance dans le processus de déploiement.

Cette approche permet aussi de s'affranchir du chef de projet comme ressource limitante dans les projets de moyenne à grande taille : les tests unitaires sont effectués indépendamment de sa charge de travail.

L'intégration requiert une mise en place significative. Il est plus facile de la faire évoluer au cours du projet, que de l'ajouter lorsque le projet est déjà bien avancé : en effet, le faisceau de contraintes qu'elle applique aux pratiques de codage est important, et faire rentrer de bonnes habitudes dans ces pratiques est un défi important.

PHP utilise beaucoup les outils écrits pour d'autres technologies comme serveur d'intégration continue. Récemment, des projets spécifiques à PHP, comme phpUnderControl et Xinc, ont vu le jour et gagnent à adopter des démarches et indicateurs spécifiques à PHP.

Outil	Description
Cruise Control	Serveur d'intégration continue écrit en Java
PhpUnderControl	Adaptation du serveur précédent aux outils PHP
Hudson	Serveur d'intégration continue, avec modules spécialisé en Java mais utilisable avec PHP
Maven	Java Server d'intégration continue, édité par la fondation Apache
Xinc	Premier serveur d'intégration continue en PHP

L'intégration continue consiste à rendre l'opération de test permanente. L'adjectif continu peut aussi s'appliquer aux autres aspects d'un projet logiciel.

La conception continue consiste à faire évoluer les documents de spécifications au fur et à mesure que les spécifications fonctionnelles sont converties en termes techniques. Cette approche est encore expérimentale, car elle perturbe beaucoup le travail de développement.

Le déploiement continue consiste à pousser en production les modifications, dès qu'elles sont archivées dans le dépôt de source. Le processus applique immédiatement les tests, et si cela fonctionne, il produit une nouvelle version.

4.3 Déploiement automatique

Le déploiement d'application est une tâche trop négligée. Souvent, elle n'est même pas explicitement prévue. La tâche elle-même est suffisamment complexe et ponctuelle pour qu'il soit naturel de confier cela à un être humain, et non pas une machine. Chaque nouvelle version entraîne des modifications particulières, sur différents aspects du logiciel (données, configurations, code PHP, structure des bases de données, etc.) qui rendent la programmation difficile à planifier et tester.

Le déploiement se fait donc le plus souvent manuellement. Il s'agit d'exporter l'application depuis le dépôt de code, de copier ses fichiers vers un serveur puis de passer les scripts de mise à jour de la base données.

Parfois, cela s'accompagne également de vidage de cache, de mise à jour de fichiers de configuration voire d'adaptation de la structure des répertoires. Le processus est long et s'allonge rapidement, pour de multiples raisons : nombre de serveurs croissant, nombre de technologies croissant, nombre de systèmes de protections croissant, etc.

Dans la pratique, il est aisé d'oublier une étape, d'en modifier l'ordre. C'est alors la catastrophe, qui demande une réparation immédiate et impérieuse, avec à la clé, une nuit blanche. De toutes ces difficultés résulte souvent un espacement des déploiements. Les mises en recette sont moins fréquentes les retours des testeurs sont donc moins rapides et il y a un ralentissement général du processus de développement.

Le coup de grâce survient quand un problème est détecté et qu'il faut mettre immédiatement en production le correctif. Est-ce que le remède n'est pas pire que le mal ?

L'automatisation est la clé de la réussite en matière de déploiement. Un processus automatisé n'oubliera pas une étape parce qu'il aura été interrompu par un mail ou un appel téléphonique. Il sera aussi capable de faire la publication de la même manière sur tous les serveurs qu'il doit gérer, et même, assurer l'installation sur les nouveaux qui sont ajoutés à la plate-forme : ce n'est plus le travail du développeur, mais de l'administrateur.

Ce processus pourra souvent être testé unitairement ou fonctionnellement afin de garantir son fonctionnement.

Le déploiement sur de multiples serveurs pourra donc se faire en parallèle en ne mobilisant qu'une personne ce qui permet un gain de temps et de ressources important. Ainsi, plusieurs personnes peuvent être formées afin de s'assurer que ce savoir-faire sera disponible à tout moment dans la société.

L'automatisation permet aussi de programmer le retour en arrière vers une version précédente. Ainsi, si le processus échoue en cours de route, il devient alors possible de demander à revenir à une version fonctionnelle du site, sans intervention humaine. C'est un gain de sécurité appréciable pour le déploiement.

Enfin, le système de déploiement automatique permet aussi d'installer des versions de tests sur différentes plate-forme cachées, pour faire tourner des tests unitaires, ou mettre à disposition rapidement des versions aux clients internes.

Il existe de nombreuses manières d'automatiser le déploiement d'une application PHP. Le moyen le plus simple est d'utiliser PHP lui-même : ce dernier est capable de fonctionner en ligne de commande et de

s'interfacer virtuellement avec toutes les technologies qu'il va lui-même utiliser sur la plate-forme Web. C'est donc une solution valide et accessible, quoiqu'un peu laborieuse à utiliser.

Un autre moyen très simple est d'écrire un script Shell contenant les commandes que nous aurions passé manuellement. Le gain fonctionnel est nul, mais nous sommes assurés que les commandes seront passées en totalité et dans le bon ordre. La sortie écran permettra également de s'assurer du succès de l'opération.

Pour procéder de manière plus professionelle, il est recommandé d'utiliser un outil d'automatisation de tâches dédié à cela. Les tâches à confier à un outil de déploiement sont des actions simples comme :

- Afficher un message ;
- Manipuler des fichiers ;
- Exporter du code depuis un dépôt Subversion ;
- Exécuter une commande en local ou via SSH ;
- Effectuer une transformation XSLT ;
- Créer une archive.

De nombreuses autres tâches sont disponibles par défaut et il est simple d'en créer de nouvelles.

Dans la liste ci-dessous, PEAR dispose d'un avantage de taille : il fournit une architecture pour la diffusion des paquets sur le réseau, à l'aide d'un système de serveur central. Un paquet PEAR est similaire à une archive mais il est possible d'y ajouter des dépendances vers d'autres paquets ainsi qu'un script qui sera exécuté après l'installation. Dans le cas d'un déploiement simple, par exemple pour une librairie ou une application peu complexe, ce moyen de distribution peut être très pratique.

Outils	Origine
Ant	L'outil le plus répandu pour les installations automatiques. Il se commande en XML, et fonctionne sans problème avec PHP.
Phing	La version PHP de l'outil précédent.
Capistrano	L'outil de référence pour Ruby.
Fredistrano	La version totalement PHP de l'outil précédent.
PEAR	Disponible avec toutes les installations PHP. Il est possible de créer aisément un canal de diffusion PEAR, interne ou public.
PHP	Très souple, pas guidant.
Phar	Archives PHP, auto exécutables.

Depuis la version 5.3, l'extension Phar est disponible par défaut dans PHP. Elle permet de créer des archives à la manière des fichiers Jar de java, ou War de J2EE. Les fichiers contenus dans une archive Phar peuvent être manipulés par PHP sans avoir besoin de la décompresser au préalable.

Au final, la vitesse et la facilité de déploiement pousse les développeurs à publier rapidement : Flickr indique qu'il publie son site jusqu'à soixante fois par semaine. Avec une publication rapide, on peut publier rapidement des correctifs importants, ou tester des micro-améliorations sans avoir à dépendre d'un processus plus important. Cela améliore considérablement la stabilité perçue de l'application. Et la répétition de l'installation améliore aussi la confiance des développeurs dans leur application.

Dans le cas de montée en charge extrême, le déploiement automatique est une arme redoutable pour pouvoir assurer des installations rapides et propres, même dans un environnement sous forte pression.

4.4 Analyse statique

La vie d'un script PHP se décompose en 2 espaces distincts : l'écriture et l'exécution. Entre les deux, il y a l'analyseur PHP, qui réalise la compilation et l'exécution du code initial, lors de la sollicitation des internautes.

L'analyse statique a pour objectif d'étudier PHP avant la compilation, pour détecter très tôt des situations à problèmes.

L'analyse s'appuie sur le code PHP avant exécution. Elle exploite des motifs de code pour reconnaître des situations : par exemple, une instruction echo ou print, appliquée à une concaténation des variables superglobales est un signe fort d'injection HTML.

L'analyse statique permet d'obtenir des résultats avant exécution, et donc sans risque pour les données ou la plate-forme de production.

L'analyse statique doit être automatisée pour pouvoir être exécutée aussi souvent que possible, et notamment tout au long du développement. De cette manière, les erreurs sont traquées lorsque le code est peu volumineux, et les incréments faciles à corriger au fur et à mesure.

L'automatisation donne aussi la maîtrise du processus de vérification aux développeurs et aux chefs de projets. Les premiers peuvent valider leur code eux-mêmes, sans passer trop de temps à implémenter les règles d'analyse ou à analyser le code PHP. C'est toujours une bonne idée que de ne pas être juge et partie dans ce type de surveillance. Pour les chefs de projets, le volume de code à analyser, dans l'absolu ou simplement après une période d'absence, a finalement peu d'impact sur les erreurs trouvées.

L'analyse statique doit être complétée par l'analyse dynamique, à l'aide de profileurs. En effet, le langage PHP est dynamique par essence : à l'exécution, il est possible de définir des classes, fonctions, variables et leur type, d'inclure ou exécuter du code PHP : tout cela modifie considérablement le fonctionnement du script si des valeurs mal intentionnées ou erronées sont introduites.

L'analyse statique doit donc affronter l'ensemble des cas possibles, alors que l'exécution d'un script donne plutôt l'aperçu du comportement lorsque des valeurs concrètes sont affectées.

Il existe quelques outils pour mesurer divers indicateurs dans le code PHP.

Outil	Objet
PHP_CodeSniffer	Analyseur de code PHP, détecteur de motifs
Pixy	Analyseur d'injections HTML et SQL
Phpcpd	Détecteur de copier-coller dans le code
PHP_Depend	Compile différents indicateurs
Phploc	Compteur de lignes de code PHP
SLOCCount	Compteur de lignes de code supportant une multitude de langages dont PHP

L'analyse statique de PHP en est à ses débuts. Il existe peu d'outils et la majorité des indicateurs sont issus des plates-formes concurrentes. Les spécificités de PHP, tels qu'une grande sensibilité aux inclusions de fichiers, sont rarement prises en compte pour des concepts issus des développements pour le bureau. Il reste donc beaucoup d'expérience à aller chercher sur le terrain.

L'analyse statique pourra notamment aller chercher des jalons naturels dans le code, telles que les structures de classes, ou fonctions, afin d'identifier les dérives dans la programmation, ou la divergence avec la conception. Par exemple, des fonctions en trop, des fonctions qui ne suivent pas les conventions de nommage, ou encore des fonctions qui ne sont pas utilisées sont autant d'informations qui doivent être gérées dans une politique générale du projet, et qui sont rarement maîtrisées, faute d'outils pour faire un rapport accessible aux intervenants du projet qui ne font pas de la programmation.

4.5 Outils de conception

La conception est souvent l'étape oubliée dans les projets en PHP : entre des spécifications souvent récurrentes, et des fonctionnalités déjà prêtes, il est trop facile de passer outre cette phase critique des projets.

Au delà de ces problèmes, les projets menés à 100 à l'heure ont tendance à repousser progressivement la conception au rang de tâches administratives sans valeur ajoutée. La conception technique, issue des spécifications, est toujours en retard sur cette dernière, quand bien même elle existe. Il y a ainsi un cercle vicieux qui s'accélère dans la vie d'un projet : moins de conception, donc moins de visibilité dans la programmation, donc plus de bogues, donc moins de temps à consacrer à

des tâches considérées comme importantes, mais trop consommatrices de temps.

L'approche du terrain est alors de faire et d'évaluer la situation après la réalisation. Cette approche est aussi issue d'assujettissement des priorités techniques aux délais de réalisation, et de la même façon conduisent à une perte de contrôle.

Bien que l'idée que la conception ne sert qu'à perdre du temps soit bien répandue, force est de reconnaître que PHP dispose de quelques outils pour faire le pont entre la conception et la production de code.

4.5.1 UML

Avec l'avènement de l'objet en PHP, il est devenu plus efficace d'utiliser des éditeurs UML pour mettre en place les classes. Si toutes les formes de communication du langage de spécifications UML ne sont pas encore maîtrisée en PHP, le langage est supporté par nombre d'entre eux :

Outil	**Objet**
ArgoUML	Produit du code PHP à partir d'un diagramme de classe, Open Source
StarUML	Open Source, uniquement sur Windows
Rational Rose	Très puissant, PHP supporté par module externe
BOUML	Éditeur très dépouillé, possibilité d'extensions

Les générateurs de diagramme permettent donc de gérer l'ensemble des classes dans des diagrammes de classes. On a une vue d'ensemble des classes et des méthodes, de leur relation, sans s'embarrasser des détails d'implémentation. Quand le projet commence à prendre de la taille, c'est un point crucial.

Une fois les diagrammes de classe établis, il est facile d'obtenir du code prêt à l'emploi, sans perdre de temps à recopier les noms des méthodes. ArgoUML produit aussi la documentation au format PHPDoc : au final, on peut donc préparer le travail de programmation dans un seul éditeur et produire automatiquement plusieurs objets qui feront gagner du temps au développement. Avantage encore, ArgoUML est capable de reprendre le code déjà produit, si jamais la conception change.

Il manque toutefois un outil de retroingénierie, qui partirait du code existant pour le transformer en diagrammes UML.

4.5.2 Modèles de données

De la même façon, les générateurs de modèle de données apportent un gain de maîtrise important.

Outil	
PowerArchitect	Open Source, interfacé avec une large variété de bases, et disponibilité d'outils de manipulations de données
MySQL WorkBench	Spécialisé pour MySQL, intensément développé par l'éditeur (MySQL AB), multi-plateforme
Toad	Très répandu dans les grands comptes, mais payant et propriétaire.

4.5.3 ORM

Les ORM sont aussi un domaine où les outils de génération de code peuvent faire gagner du temps. Il s'agit des Object-Relationnal Mappings, c'est à dire des correspondances entre les classes PHP et les tables SQL. Il existe en effet une similitude entre les deux qui rend naturel un rapprochement : un nom de colonne devient une propriété d'un objet, et une table est associée à une classe.

Ce type d'approche est efficace si les objets métiers sont représentés par des tables simples en base, ce qui est souvent le cas dans les applications Web. On en vient donc à produire souvent les mêmes fonctionnalités de correspondances, sans plus-value.

Il y a trois approches :

- l'ORM se base sur la base de données et en déduit des classes PHP (Propel) ;
- L'ORM lit un fichier de configuration indépendant et en déduit les classes PHP ;
- L'ORM adapte la base de données aux classes PHP produites.

C'est surtout les deux premières approches qui sont utilisées, car la base de données reste la chasse gardée des administrateurs qui n'autorisent pas facilement aux scripts PHP de modifier les modèles. De plus, les clients comprennent bien cette notion de modèle de données, et c'est

aussi un moyen de discussion avec eux, alors que les classes PHP sont plus difficiles à lire ou à comprendre.

La troisième approche est plutôt adaptée aux entrepôts clé-valeur, de type CouchDB.

Outil	**Méthode**
Doctrine	Extrait les objets d'un fichier de description
Propel	Extrait les objets de la base de données
EzPDO	Extrait les objets du code PHP
CouchDB	Pas besoin d'ORM

4.5.4 Documentation technique

La documentation technique est la documentation interne du projet, elle est destinée aux développeurs. La plupart du temps, ce type de documentation est totalement absent d'un projet, ce qui handicape malheureusement beaucoup la prise en main du projet par un nouveau collaborateur, ou par un prestataire externe.

La documentation technique doit être de deux types : de référence, et globale.

La documentation de référence est le catalogue de toutes les classes, méthodes et fonctions qui ont été mises en place dans le code. La pratique recommandée ici est d'utiliser la syntaxe JavaDoc, directement dans le code. Lorsque cette documentation est présente, plusieurs outils de documentation sont capables d'analyser le contenu et de produire un ouvrage avec toutes les informations, ainsi que les indexes croisés adaptés pour naviguer rapidement dans l'ouvrage.

La documentation de référence doit être complétée par une présentation globale de l'architecture de l'application. En effet, la référence tend à donner une vue morcelée de l'application, en se concentrant sur les briques élémentaires. Il faut donc compléter la vue par une description littérale du fonctionnement de l'application, afin de placer les grosses masses.

Outil	Description
PhpDocumentor	Outil de référence dans le monde PHP pour la génération de documentation technique au format JavaDoc

Les outils de conception permettent d'anticiper de nombreux problèmes et de donner des guides importants lors de la phase de développement. UML est un standard en PHP, comme dans le reste du monde informatique, et à l'heure actuelle, seuls les diagrammes de classes sont réellement exploités.

Les informations de conceptions aident aussi à suivre la qualité du projet, en fournissant un référentiel par rapport auquel se comparer pour savoir si la programmation suit le fil du projet. C'est aussi un outil important pour la planification du projet : à partir du diagramme UML, on peut découper les classes en tâches, vérifier que l'ordre de développement est cohérent pour pouvoir les tester ou les imbriquer et obtenir un premier rendu visible. Cela a donc un impact direct sur la gestion du chef de projet, et la mesure du délai de livraison.

Le besoin de documentation technique s'atrophie de lui-même lorsque la conception prend le relais : à l'aide de ces spécifications, il est possible de produire le code et la documentation associée, pour préparer le travail des développeurs, et les soulager par la même de cette tâche.

4.6 Méthodes de programmation

Convertir les besoins fonctionnels d'un client en spécifications techniques est un défi. De nombreuses connaissances à la fois métier et techniques sont nécessaires. Pourtant, les méthodes traditionnelles de développement (modèle en cascade, cycle en V, etc.) prônent la rédaction de spécifications fonctionnelles complètes avant de commencer la programmation. Celles-ci doivent décrire dans le détail chaque aspect de l'application à créer et replacer chaque fonctionnalité dans la globalité de l'application. La rédaction nécessite une forte capacité d'abstraction et beaucoup de rigueur pour n'oublier aucun aspect du besoin. Des mois sont souvent nécessaires pour aboutir au document final.

A partir de là, la programmation commence. Suivant sa durée, cela produit l'effet tunnel : le client a livré ses spécifications, mais n'obtient aucun résultat concret avant la phase de recette. Le développement progresse, mais comme l'application n'est pas fonctionnelle, il n'y a rien à voir, ni a évaluer. Au bout du tunnel, le client attend la surprise : elle sera bonne ou mauvaise. Outre les raisons évidentes d'exécution et de compétences, d'autres raisons peuvent avoir un impact important lors de

la recette : si le développement est trop long, il arrive tout simplement que le besoin évolue durant le développement.

Les applications sont de plus en plus complexes, de plus en plus longues, et les besoins des clients sont de moins en moins précis : pour conserver le lien entre les utilisateurs et les développeurs, de nouvelles approches méthodologiques ont émergé.

Des méthodes plus réactives ont été inventées, avec notamment des cycles de développement plus courts. Elles sont rassemblées sous le nom de méthodes agiles, dont les principaux représentants sont Scrum et l'eXtreme Programming (XP). Ces deux méthodes abordent les choses sous deux angles différents et, loin de se concurrencer, ont plutôt tendance à se compléter.

Ces méthodes sont destinées à des équipes de petites tailles.

Sur des projets de grande envergure, il est parfois possible de scinder le projet en plusieurs petites équipes, avec une équipe de coordination.

4.6.1 Scrum

Scrum a été développée au début des années 1990. Cette méthode est adaptée à la gestion de projets de tout type et pas seulement informatiques. Son fonctionnement se base sur une terminologie précise. Scrum définit trois rôles principaux :

- **le directeur de produit** : il est le représentant des clients et utilisateurs. C'est lui qui définit les priorités fonctionnelles et qui oriente le projet ;
- **le ScrumMaster** : il est chargé de protéger l'équipe de tous les éléments extérieurs et de résoudre ses problèmes non techniques. Il veille également au respects des valeurs de Scrum ;
- **l'équipe** : ce sont les développeurs, graphistes, intégrateurs et tout autre poste technique du projet. Il n'y a pas de hiérarchie au sein de l'équipe et elle est autogérée. C'est-à-dire que ses membres s'organisent à leur guise pour le bien du projet.

Des intervenants peuvent également s'intégrer au projet de façon plus ponctuelle. Ils souhaitent suivre le projet sans y participer directement.

Au lancement du projet, l'équipe découpe le besoin fonctionnel en tâches auxquelles le directeur de produit attribut un certain nombre de points. Ces points n'ont pas d'unité. Leur valeur est relative à la vitesse de travail de l'équipe. La planification est itérative dans Scrum. Chaque itération est appelée sprint et dure entre 2 et 4 semaines. Au début du projet, l'équipe estime combien de points elle peut réaliser par sprint.

Au début de chaque sprint, le directeur de produit indique quelles tâches il souhaite voir réaliser pendant celui-ci en tenant compte des points associés à chaque tâche afin de ne pas dépasser la capacité de travail de

l'équipe. En fin de sprint, on calcule le nombre de points réalisés par l'équipe et on en calcule la moyenne par sprint. C'est ce qu'on appelle la vélocité. Elle sert à estimer de manière assez réaliste la capacité de travail de l'équipe.

L'application est livrée au directeur de produit à chaque fin de sprint pour qu'il puisse la tester. A ce stade, elle est incomplète en termes de fonctionnalités, mais elle est normalement utilisable. Le directeur de produit a donc très régulièrement une version testable de l'application. Il peut ainsi signaler très vite d'éventuels bogues ou écarts fonctionnels.

Une autre conséquence de ce fonctionnement est de laisser au client la possibilité de se tromper. C'est-à-dire qu'il peut s'apercevoir en testant l'application que le comportement demandé n'est pas adéquat. Il pourra donc demander à le rectifier dans l'itération suivante. Comme le client se rend compte très tôt des problèmes, cela réduit les coûts de correction et ne risque pas de faire échouer le projet in fine.

A chaque fin de sprint, le client peut décider que l'application lui convient et que le niveau de fonctionnalité attendu est atteint, même s'il reste des tâches à accomplir. En effet, il peut décider à tout moment de la pertinence des points. Quand il n'y a plus de points, le projet s'arrête.

La méthode Scrum ne couvre aucune technique d'ingénierie logicielle. Son utilisation pour le développement d'une application informatique nécessite donc de lui adjoindre une méthode complémentaire comme l' eXtreme Programming.

4.6.2 eXtreme Programming

L'eXtreme Programming (XP) est un ensemble de valeurs et de pratiques qui visent à faciliter les développements informatiques. Cette méthode reprend des principes existants, mais les pousse à l'extrême, d'où son nom.

Les valeurs de l'eXtreme Programming sont :

- La communication ;
- La simplicité ;
- Le feedback ;
- Le courage ;
- Le respect.

Ces valeurs vont dans le sens d'un développement apaisé au sein d'une équipe ouverte et intelligente. Elles se déclinent en treize pratiques qui se renforcent mutuellement :

- **Client sur site** : pour assurer une grande réactivité le client doit idéalement être sur le site où travaille l'équipe. Si ce n'est pas possible, quelqu'un peut le représenter **;**

- **Jeu du Planning** : le client crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les implémenter. Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible ;
- **Intégration continue** : afin de détecter au plus tôt d'éventuels défaut, les nouveaux développements sont intégrés à l'application très régulièrement ;
- **Petites livraisons** : les livraisons doivent être les plus fréquentes possible ;
- **Rythme soutenable** : le développement est un exercice intellectuel créatif qui demande d'être en forme. On veillera donc à garantir un rythme de travail soutenable et notamment à ne pas faire faire d'heures supplémentaires à l'équipe deux semaines d'affilée ;
- **Tests fonctionnels** : à partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement ;
- **Tests unitaires** : l'équipe rédige au fur et à mesure des tests validant le comportement technique de l'application ;
- **Conception simple** : cette méthode encourage la simplicité de conception afin de ne pas introduire de complexité inutile dans l'application ;
- **Utilisation de métaphores** : on évite le jargon technique ou métier car l'équipe et le client se comprennent beaucoup mieux en parlant un langage simple ;
- **Remaniement du code** : amélioration régulière de la qualité du code sans en modifier le comportement. Cette valeur est intimement liée aux tests unitaires et fonctionnels ;
- **Responsabilité collective du code** : l'ensemble de l'équipe est responsable du code. Personne n'a la propriété exclusive d'une partie de l'application ;
- **Convention de programmation** : afin de faciliter le travail collaboratif, il est important de définir des règles communes ;
- **Programmation en binome** : les développeurs travaillent à deux sur un même poste. L'un tape le code tandis que l'autre dialogue avec lui afin de concevoir à deux le code. Les binomes changent régulièrement pour faciliter la communication et le partage des connaissances au sein de l'équipe.

Certaines des pratiques de l'eXtreme Programming peuvent sembler radicales, mais elles ne sont souvent que du bon sens poussé au maximum.

De prime abord, les méthodes agiles sont déroutantes car elles remettent en cause de nombreuses habitudes : elles demandent même un certain courage de la part des développeurs. Il faut accepter de livrer le fruit de son travail et donc une partie de soi au regard et à l'appréciation des autres. Cela peut être difficile, mais cela pousse également à l'excellence.

Les méthodes agiles ne sont pas forcément adaptées à tous les projets, ni à toutes les entreprises. Certaines pratiques peuvent rendre difficile voire impossible leur adoption comme :

- un blocage culturel devant l'absence de hiérarchie et l'autonomie accordée à l'équipe ;
- des équipes de taille trop importante ;
- un client peu disponible ;
- la résistance au changement.

Enfin, le principe même des méthodes agiles rentre souvent en conflit avec l'aspect commercial d'un projet. Comment chiffrer un projet dont on ne connaît pas précisément le périmètre fonctionnel final ? La facturation en mode forfait est peu adaptée à ce genre de méthode. Il faut que le client accepte un fonctionnement au temps passé, au prix d'une moindre visibilité sur le budget final.

4.7 Maîtrise de la qualité du code

Trop souvent, la qualité du code est vérifiée après programmation : la raison avancée pour cela est qu'il n'y a pas moyen de faire de vérification tant que la programmation n'est pas faite. La solution consiste alors à faire appel à un expert externe au projet, pour relire le code, identifier des points de non-qualité et suggérer des corrections. Ces dernières touchent parfois à la conception, parfois au code, ou encore à des pratiques de la programmation : la charge de correction fait alors qu'il est alors trop tard pour les appliquer.

La première réaction est alors de relancer l'audit dès la prochaine étape importante du projet : nouvelle version majeure, nouvel ajout important, nouvelle fonctionnalité stratégique. Le rythme est alors pris mais les interventions sont toujours en retard. Il faut prendre le problème beaucoup plus en amont pour pouvoir assurer une qualité continue et prévisible.

Pour faire progresser la qualité de la programmation, deux ingrédients essentiels sont nécessaires :

- Une référence, qui porte le nom de "Mantras PHP" ;
- Un processus de surveillance, qui est baptisé "audits croisés".

Les mantras PHP sont une liste courte de recommandations de qualité à appliquer au code PHP lors de la programmation. Ces recommandations peuvent être de plusieurs types, et doivent être entre 5 et 7 au maximum.

L'objectif est de proposer un référentiel simple et facile à mémoriser, pour que chaque développeur les connaisse bien. L'analogie est faite avec les mantras (ou le chapelet) où les mêmes litanies sont répétées : ici, le développeur doit les ressasser en même temps qu'il produit du code PHP. Les mantras PHP sont différents des guides de références : ces derniers sont trop littéraires ou trop volumineux pour être applicables facilement durant la production de code.

Voici quelques idées de mantras PHP :

- Toujours valider les données en entrées;
- Toujours protéger les données en sortie ;
- Vérifier l'application des conventions de programmation ;
- Utiliser un dossier hors Web ;
- Éviter de lire plus de 100 lignes dans la base de données;
- Ne pas mettre de tableaux dans les sessions ;
- Éviter les "ereg".

Idéalement, cette liste doit avoir été créée avec les développeurs, afin de produire un référentiel auquel ils pourront s'identifier facilement et qu'ils comprendront aisément. Le mieux est de rédiger cette liste lors d'une réunion de remue-méninges commun, puis de publier cette liste de manière très visible : fond d'écran, impression à coté de l'écran, ou à la machine à café.

Une fois bien connus des développeurs, les mantras PHP sont alors la référence à utiliser pour évaluer la qualité de chaque production de code. On peut s'en servir dans un audit croisé, ou revue de code, pour vérifier que les mantras PHP sont bien appliqués.

Les audits croisés consistent à organiser les développeurs en binomes (ou éventuellement en trinomes) : avant chaque archivage de code dans le dépôt, le binome d'un développeur doit relire le code qui va être publié, et s'assurer que les PHP Mantras ont bien été appliqués. Il doit alors identifier les points à corriger, et les faire modifier par l'auteur initial.

Les audits croisés ont de multiples avantages :

- Le chef de projet ou l'expert ne sont pas une ressource limitante, puisque l'audit ne passe pas forcément par eux. Ils peuvent évidemment apporter un regard supplémentaire et différent aux audits ;

- Les audits croisés dépassent les cloisonnements des projets : tant qu'un développeur connaît suffisamment PHP, il peut être mis en binome avec un autre. Cela permet de créer un espace commun et entièrement technique et d'éviter la fragmentation des connaissances techniques ;
- Le rythme des audits peut être facilement adapté à celui du projet : en l'absence de développeurs, les audits peuvent être réorganisés pour s'assurer que le code a bien été relu.
- L'effet principal de cet organisation est de montrer clairement les objectifs de qualité et de rappeler aux développeurs qu'une vérification sera effectuée inévitablement ;
- Il est recommandé de mettre en binome un développeur sénior et un développeur junior : le premier sera mieux capable d'assimiler les suggestions originales d'un nouveau au projet et de guider le nouveau dans les mantras locales, sans le noyer dans trop de règles.

Il n'existe actuellement aucune référence de mantras PHP. Toutefois, la bibliographie autour de ces sujets propose suffisamment de recommandations et les développeurs disposent de suffisamment d'expérience pour savoir être force de proposition.

Pour les audits croisés, il est apparu récemment des outils de revue collaborative : l'objectif est de fournir un dépôt central pour relire le code, l'annoter, et prévenir directement l'auteur des portions de code visée. Ce type d'outil doit pouvoir s'interfacer avec les dépôts de code et prendre en compte les différentes versions des fichiers.

Plusieurs outils existent actuellement :

Outil	**Avantage**
Groogle	Open Source, écrit en PHP et pour PHP
Review Board	Simple et universel, écrit en Python
Rietveld	Écrit par Google, fonctionne sur Google Apps
SmartBear	Logiciel commercial

Le principe de base des audits croisés est d'accélérer les audits qui se font généralement à grands frais et trop rarement. On arrive aussi à remonter

le cycle de la vie du projet en plaçant la création des mantras PHP dès le début du projet, voire même avant la conception de l'application.

Par la simple application de la discipline du code, on élève nettement la conscience dès l'écriture du code et on obtient en bout de compte un niveau de qualité supérieur. Au passage, les équipes gagnent en cohésion et en niveau de collaboration.

4.8 Implication des utilisateurs

Le volume des interactions avec les utilisateurs est un aspect primordial dans les projets Web : cela demande généralement du temps, du temps et encore du temps. La notion d'utilisateur couvre plusieurs personnes :

- Les clients, qui expriment leurs besoins ;
- Les vérificateurs, qui identifient un dysfonctionnement ;
- Les utilisateurs, qui souhaitent simplement tirer profit de l'application.

La plupart du temps, ce sont les mêmes personnes qui portent ces mêmes casquettes tout au long du projet. Ils sont à l'origine du projet ou ils fournissent des documents sous le nom de spécifications ; ils sont aussi utilisateurs et demandent des documentations, un suivi assez lâche du projet et remontent des problèmes, que ce soit dans le cadre d'une recette ou d'une utilisation routinière.

En fait, la communication avec les utilisateurs est toujours de même nature, mais comme elle porte des noms différents suivant les phases du projets, elles sont souvent dissociées. Par exemple, un bogue et une spécification sont finalement des prescriptions identiques, mais dans des circonstances différentes : le premier intervient après livraison, et représente une erreur du développeur. Le second est émis au début du projet et ne se compare pas à une réalité de programmation. De la même façon, un souci de documentation engendre un bogue, ou une nouvelle spécification.

Une fois cela remarqué, le volume et la qualité de la communication entre les utilisateurs et l'équipe de développement subit une accélération permanente, dès le début du projet. A ce moment, la communication est bourrue, et se résume souvent à des échanges rapides et oraux. Les spécifications sont parfois livrées à l'avance, et relues par les développeurs, avec parfois l'accompagnement des utilisateurs. Comme le projet est encore à un stade immature, la communication se fait par voie orale, avec un nombre d'intervenants très limité : peu de développeurs, peu d'utilisateurs. C'est le règne des communication instantanées, comme le téléphone ou les messageries instantanées, avec très peu d'archivage.

Puis, avec les premières livraisons, apparaissent les premiers volumes : il faut livrer une documentation pour expliquer comment fonctionne l'application, assurer le suivi des bogues, et augmenter les spécifications de nouveaux éléments ou de rétroactions. Cette phase se fait souvent par mail, pour conserver l'aspect réactif des premières communications : la communion entre les développeurs et les clients est toujours de mise, mais elle a maintenant pris un caractère supplémentaire : elle est traçable. Cela reste valable pour de petites équipes et des dossiers peu complexes.

Les systèmes par mail ou messagerie instantanée restent des outils immédiats et cantonnent les échanges à des relations directes (asynchrones, mais directes). L'étape suivante est de disposer d'un outil de centralisation pour organiser plusieurs nouveaux aspects :

- Différencier les bogues des spécifications (nouvelles ou réitérées) ;
- Capitaliser sur les expériences précédentes ;
- Etablir les priorités des tâches et savoir lesquelles sont terminées, sans interroger les développeurs ;
- Assurer des communications canalisées autour de sujets précis.

C'est à ce stade que les applications de gestion de bogues interviennent. Dans leur version minimale, elles offrent une interface Web où se retrouvent développeurs et utilisateurs, pour décrire des points particuliers de l'application et suivre les modifications qui s'en suivent.

Les versions les plus sophistiquées s'interfacent avec le dépôt de code, pour marquer la version dans laquelle les modifications ont été apportées, et ainsi transmettre automatiquement aux utilisateurs les versions de publications qui supporteront leurs demandes ; elles s'interfacent aussi avec les outils de tests unitaires ou fonctionnels pour faire correspondre une fonctionnalité à un ou plusieurs tests spécifiques. Cette dernière approche pousse à l'utilisation des tests unitaires, et à une capitalisation technique des demandes de corrections.

Outil	Description
Mantis	Probablement l'outil PHP le plus répandu. Un peu austère, mais très adaptable à nombre de situations.
Jira	Gestion fine des équipes et des utilisateurs ; configuration plus facile. Propriétaire ; interfacable avec GreenPepper ou les outils Atlassian.
Trac	Testé sur de nombreux projets, même non-PHP. La gestion de bogues y est assez austère.
Bugzilla	L'outil de gestion des bogues le plus connu dans le monde Open Source. Puissant mais assez spartiate.

Les aspects documentations sont aussi à prendre en compte : outre les bogues, qui signalent un dysfonctionnement, il est aussi important de mettre à disposition des utilisateurs des explications écrites. Il y a alors deux types de documentations qui sont produits : les documentations de référence et les manuels.

Les documentations de référence sont générées automatiquement à partir du code PHP. Elles sont intéressantes uniquement pour les développeurs, et si la prise en main de l'application est simple. En effet, elles présentent en détails les briques disponibles, mais pas la manière de les utiliser.

Les manuels empruntent aux documentations de référence, mais ajoutent des tutoriels, qui assistent les utilisateurs à l'utilisation d'une fonctionnalité. Ils sont clairs et faciles à aborder pour un néophyte, mais cantonnent les experts dans des rails fixés à l'avance.

Dans les deux cas, les manuels sont à produire initialement par les développeurs. En effet, comme ce sont ces derniers qui ont produit l'application, ce sont les mieux placés pour décrire le fonctionnement. L'exemple typique est la documentation PHP, qui est organisée en fichiers XML, produite à la base par les auteurs des fonctionnalités, puis prise en charge par l'équipe de documentation, pour les traductions, les corrections et les ajouts.

Toutefois, c'est un travail laborieux pour les équipes et il est difficile de faire passer le message aux utilisateurs, qui n'évoluent pas dans le même monde. Toujours dans le cas de la documentation, il est souvent répété que les notes de bas de page sont souvent plus utiles aux utilisateurs que la documentation elle-même. En fait, c'est surtout que les utilisateurs se heurtent aux même problèmes et qu'il est crucial d'organiser la communication entre eux, pour profiter d'une expérience qui est totalement étrangère aux développeurs.

Ainsi, à partir de la documentation officielle, il est souvent intéressant d'ouvrir un système d'annotation ou de commentaires, destiné à faire participer la communauté. Au fil de l'augmentation de la taille de la communauté, on peut passer à un niveau de wiki et remettre aux développeurs la charge d'un blog. Le blog sert à communiquer les nouveautés à l'ensemble de la communauté et les utilisateurs prennent le relais.

L'avantage de remettre la documentation aux utilisateurs est qu'ils utilisent une approche très appliquée à un objectif (i.e. « j'utilise strip_tags pour limiter les possibilités d'injection HTML »), alors que la fonctionnalité a été pensée souvent dans un esprit très généraliste («strip_tags permet de retirer les balises d'une chaîne»).

La transition ne se fait pas naturellement et il est important d'avoir une communauté suffisamment large pour susciter des vocations à la gestion de la documentation. Malgré sa qualité reconnue, la documentation PHP n'est gérée que par une petite centaine de personnes, alors qu'il y a plus de 17 traductions et 250 miroirs à travers le monde.

Outil	**Description**
MediaWiki	Le wiki de Wikipedia, facile à maîtriser, et à faire monter en puissance
Kmwiki	Un Wiki optimisé pour la gestion des connaissances

Au delà des wikis, la transition de la documentation vers la communauté peut prendre d'autres formes encore : écriture de tutoriels, livres, conférences, podcast, vidéocast, rencontres de groupe d'utilisateurs, ouverture d'une API publique, etc.

Les utilisateurs doivent aussi prendre en charge une partie des tests unitaires et fonctionnels : c'est à la fois un outil de communication avec les développeurs, d'assurance qualité et de documentation. Certains projets établissent un pont entre l'application et la documentation à l'aide de tests, ce qui permet de vérifier que la documentation est synchrone avec le développement. De plus, l'approche métier des utilisateurs est cruciale pour que l'application atteigne sa propre utilitée.

5 Une nouvelle frontière

L'industrialisation est une nouvelle frontière pour PHP. C'est un domaine qui a rarement été abordé jusqu'à présent, sauf à mettre en place des outils spécifiques à une organisation particulière. Il s'agit désormais d'organiser les pratiques autour de méthodes et d'outils.

5.1 PHP n'a pas encore exprimé son identité

Plusieurs caractéristiques techniques de PHP sont rarement prises en compte :

- PHP adopte la philosophie "Share Nothing" : il essaie au maximum de fonctionner sans avoir à coordonner ses processus : ils fonctionnent de manière autonome et indépendante.
- PHP fonctionne au hit, avec une durée de vie très courte : les opérations de nettoyage de mémoire sont reportées à l'extinction.
- L'importation de larges bibliothèques de composants est inutile en PHP, voire nocive. Cette importation doit être faite pour chaque sollicitation de script et, avec le trafic, crée des goulets d'étranglement. Les politiques d'importation de composants doivent donc être étudiées avec beaucoup de soin.
- La nature modulaire de PHP permet de faire évoluer la plate-forme en intégrant des bibliothèques écrites en C ou C++, et d'ouvrir les API directement dans les scripts PHP.
- PHP est indissociable du serveur Web qui l'héberge, et qui gère son existence et son environnement.

Toutes ces caractéristiques doivent être prises en compte pour bien comprendre le fonctionnement des scripts et les maîtriser.

5.2 PHP n'exploite pas encore ses capacités de collaboration

PHP est né sur le Web, pour le Web. Dès l'origine, c'est une plate-forme pour communiquer qui a été mise en place.

PHP sait réaliser l'interface entre des serveurs Subversion, HTTP, SSH, FTP, ou encore manipuler des documents XML, XMI ou PDF. Il sert souvent de plaque tournante entre des mondes très différents et cela s'applique complètement aux notions d'industrialisation.

Par ailleurs, écrire une application en PHP revient à la rendre accessible à quiconque dispose d'un navigateur Web. Dès que ces données sont accessibles, on peut faire collaborer les clients, les chefs de projets, les utilisateurs. Ce type d'approche, comparable aux projets SourceForge ou Gforce, donne immédiatement une dimension d'équipe aux solutions retenues.

Pour aller plus loin, on peut aussi imaginer que les collaborateurs ne soient plus regroupés uniquement par projet, mais bien par métier : les équipes de tests collaborent de manière transversale pour multiplier les tests d'encadrement, ou encore les utilisateurs pour améliorer leur pratique des documentations collaboratives.

5.3 Des idées à explorer

De nombreux outils restent à mettre en place pour assurer le bon développement des pratiques d'industrialisation :

- Outil de déploiement : branché dans le dépôt de source, ces outils, adaptés aux technologies connexes de PHP, permettront de publier du code source de manière atomique, de gérer les configurations de production et préproduction.
- Un outil d'audit collaboratif, avec une analyse statique du code, et des interfaces personnelles pour commenter le code, ou modifier le statut des points identifiés. Les motifs recherchés dans le code pourraient aussi être ouverts à tous, pour couvrir le maximum de pratiques. Cet outil pourrait aussi produire des rapports d'analyse au format PDF/ODT, pour diffusion auprès des clients.
- Un outil d'intégration continue, écrit en PHP, tel que Xinc, mais avec plus d'expérience.
- Une pile d'industrialisation, qui rassemblerait les outils allant de la conception à la mise en production, en passant par l'analyse statique et dynamique. Cette pile proposerait une interface commune pour les différents outils et méthodes d'industrialisation, accessible facilement via le Web ou l'intranet.

- Une interface entre les tests fonctionnels et la documentation : produire et vérifier la documentation à partir des tests unitaires et fonctionnels.
- Le lien entre les composants d'un projet Web et la planification des tâches, afin de fournir la date prévue de fin d'un projet.
- Une bibliothèque de référence de Mantras PHP, avec des listes de mantras à choisir, et, le cas échéant, des outils pour identifier les violations.

Tout ces outils pourront être développés par des éditeurs indépendants ou des communautés Open Source.

5.4 La communauté est un atout majeur

La communauté PHP compte aujourd'hui plus de 5 millions de développeurs ; plus de 10 000 projets en cours ; une centaine de livres et plus de 2 conférences mensuelles dans le monde. La communauté est organisée en groupe d'utilisateurs autonomes, présents dans chaque pays, ou presque.

Cela est dû à une grande facilité d'accès à la technologie elle-même, et aussi à un vif esprit de solidarité et de collaboration. Les forums et les listes de diffusion assurent une circulation rapide des informations et un relais pour les projets et produits.

5.5 Les développeurs vont gagner en discipline

Les développeurs PHP sont souvent autodidactes et apprennent la technologie comme ses pratiques au fur et à mesure de leur expérience. Or, s'il est rapide d'apprendre à utiliser PHP par une méthode expérimentale et un peu de bon sens, il est plus long de découvrir les avantages de certaines techniques. La gestion de projets informatiques est longue à venir.

Les audits signalent souvent une dérive progressive dans l'application des bonnes pratiques. Ces dernières sont souvent connues des développeurs, mais aussi souvent mises à l'écart, par simple manque de visibilité sur les retours qu'elles engendrent. On voit souvent la documentation, les tests, la conception ne pas résister aux pressions des clients et progressivement céder la place au chaos. Ces outils sont justement là pour éviter ces situations.

A l'avenir, il faut que les développeurs apprennent à résister à ces pressions et sachent mettre en avant les avantages même des clients et utilisateurs : loin de les pénaliser, les bonnes pratiques sont capables de

les faire gagner en crédibilité et en efficacité, pour peu qu'elles soient simplement appliquées.

6 Bibliographie

Portails sur l'industrialisation PHP

- PHP to the moon : http://phptothemoon.com/
- Industrialisation PHP : http://www.industrialisation-php.com

Outils d'audits de code

- CodeSecure : http://www.armorize.com/
- RATS : http://www.fortify.com/security-resources/rats.jsp
- SWAAT : http://www.owasp.org/index.php/Category:OWASP_SWAAT_Project

Conventions de code

- PEAR coding standard : http://pear.php.net/manual/fr/standards.php
- PHP coding standard : http://www.dagbladet.no/development/phpcodingstandard/
- Zend Framework coding standard : http://framework.zend.com/manual/en/coding-standard.html

Veille technologique

- Delicious : http://delicious.com/
- Digg : http://www.digg.com/
- Diigo : http://www.diigo.com/

Dépôts de code

- Baazar : http://bazaar-vcs.org/
- CVS : http://www.nongnu.org/cvs/
- Git : http://git-scm.com/
- Mercurial : http://mercurial.selenic.com/wiki/
- Perforce : http://www.perforce.com/
- Rational ClearCase : http://www-01.ibm.com/software/awdtools/clearcase/
- Subversion : http://subversion.tigris.org/
- Comparison of revision control software : http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

Frameworks

- Symfony : http://www.symfony-project.org/
- Zend Framework : http://framework.zend.com/

- Liste de frameworks
  PHP : http://fr.wikipedia.org/wiki/Liste_de_frameworks_PHP

IDE

- Aptana : http://www.aptana.com/
- Eclipse : http://www.eclipse.org/
- Komodo : http://www.activestate.com/komodo/
- NetBeans : http://www.netbeans.org/
- PDT : http://www.eclipse.org/pdt/
- Zend Studio : http://www.zend.com/fr/products/studio/

Débogueur

- APD : http://apd.communityconnect.com/
- DBG : http://www.php-debugger.com/dbg/
- Xdebug : http://www.xdebug.org/

Tests unitaires

- PHP Unit : http://www.phpunit.de/
- SimpleTest : http://www.simpletest.org/
- phpt : http://qa.php.net/write-test.php

Intégration continue

- Under control : http://cruisecontrol.sourceforge.net/
- PhpUnderControl : http://phpundercontrol.org/about.html
- Hudson : https://hudson.dev.java.net/
- Maven : http://maven.apache.org/
- Xinc : http://code.google.com/p/xinc/

Déploiement automatique

- Ant : http://ant.apache.org/
- Capistrano : http://www.capify.org/
- Fredistrano : http://code.google.com/p/fredistrano/
- Phing : http://phing.info/trac/
- PEAR : http://pear.php.net/

Analyse statique

- PHP Beautifier : http://pear.php.net/package/PHP_Beautifier
- PHP_CodeSniffer : http://pear.php.net/package/PHP_CodeSniffer
- Pixy : http://pixybox.seclab.tuwien.ac.at/pixy/
- RATS : http://www.fortifysoftware.com/security-resources/rats.jsp
- phpcpd : http://github.com/sebastianbergmann/phpcpd
- phploc : http://github.com/sebastianbergmann/phploc
- PHP_Depend : http://pdepend.org/news.html

Modélisation UML

- ArgoUML : http://argouml.tigris.org/
- StarUML : http://staruml.sourceforge.net/
- Rational Rose : http://www-01.ibm.com/software/rational/
- BOUML : http://bouml.free.fr/

Modélisation de base de données

- Mogwai : http://mogwai.sourceforge.net/
- Navicat : http://www.navicat.com/
- Toad : http://www.toadsoft.com/
- MySQL WorkBench : http://dev.mysql.com/workbench/
- Power Architect : http://www.sqlpower.ca/page/architect

Issue trackers

- Jira : http://www.atlassian.com/software/jira/
- Trac : http://trac.edgewall.org/
- Redmine : http://www.redmine.org/
- Mantis : http://www.mantisbt.org/
- Bugzilla ; http://www.bugzilla.org/

Documentations

- KmWiki : http://kmwiki.wikispaces.com/
- Mediawiki : http://www.mediawiki.org/wiki/MediaWiki/fr
- DocBook : http://www.docbook.org/

7 Licence et diffusion

7.1 OpenContent License (OPL)

Terms and Conditions for Copying, Distributing, and Modifying.

Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent (OC) as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the OC a copy of this License along with the OC. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the OC for use offline, you may at your option offer instructional support for the OC in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge a fee for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY

4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU. SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE OC AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE OC, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7.2 Diffusion

Ce document est publié sous licence Open Content de manière à favoriser sa diffusion. Vous pouvez donc utiliser ce document librement à condition de mentionner clairement les auteurs «Damien Seguy et Jean-Marc Fontaine pour Alter Way», un lien vers le site original de ce document, (http://www.alterway.fr/) et de garder le caractère «Open Content» de ce dernier.

Le texte de la licence open content est disponible ci-dessus en version originale, http://opencontent.org/opsl.shtml

Seule la version originale peut être utilisée à des fins légales.

Publié sous licence Open Content, ce document peut être copié et diffusé autant de fois que vous le désirez.