



OpenSAMM



Identifier et réduire le risque durant
le cycle de développement logiciel

Antonio Fontes
OWASP - Suisse

Training Day Paris



OWASP France

26 avril 2011

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation
<http://www.owasp.org>

Infos intervenant

- Antonio Fontes
- 6 années d'expérience dans la sécurité « logicielle »
- Directeur de la société  L7 Sécurité (Genève, CH)
- Spécialités:
 - Défense des applications web
 - Développement sécurisé
 - Tests d'intrusion
 - Modélisation de menaces / estimation, qualification du risque
- OWASP:
 - Comité OWASP Suisse 
 - Responsable OWASP Suisse romande



Programme

- Situer le contexte et le besoin
- Survol des alternatives
- OpenSAMM:
 - Présentation
 - Domaines d'activité
 - Activités de sécurité
 - Scénarios de déploiement
 - Evaluation
- L'approche initiale
- Ecueils
- Conclusion, vos questions



Quel besoin, pour quel contexte?



Constat

- Multiplication des application web au sein des organisations
- Multiplication des technologies
- Interconnexion des services stratégiques (backoffice/mainframe)
- Multiplication des unités de développement:
 - Nombreux cycles de développement à l'oeuvre
 - Externalisation croissante du cycle ou partie



Le paradigme « pentest »

analyse

conception

implémentation

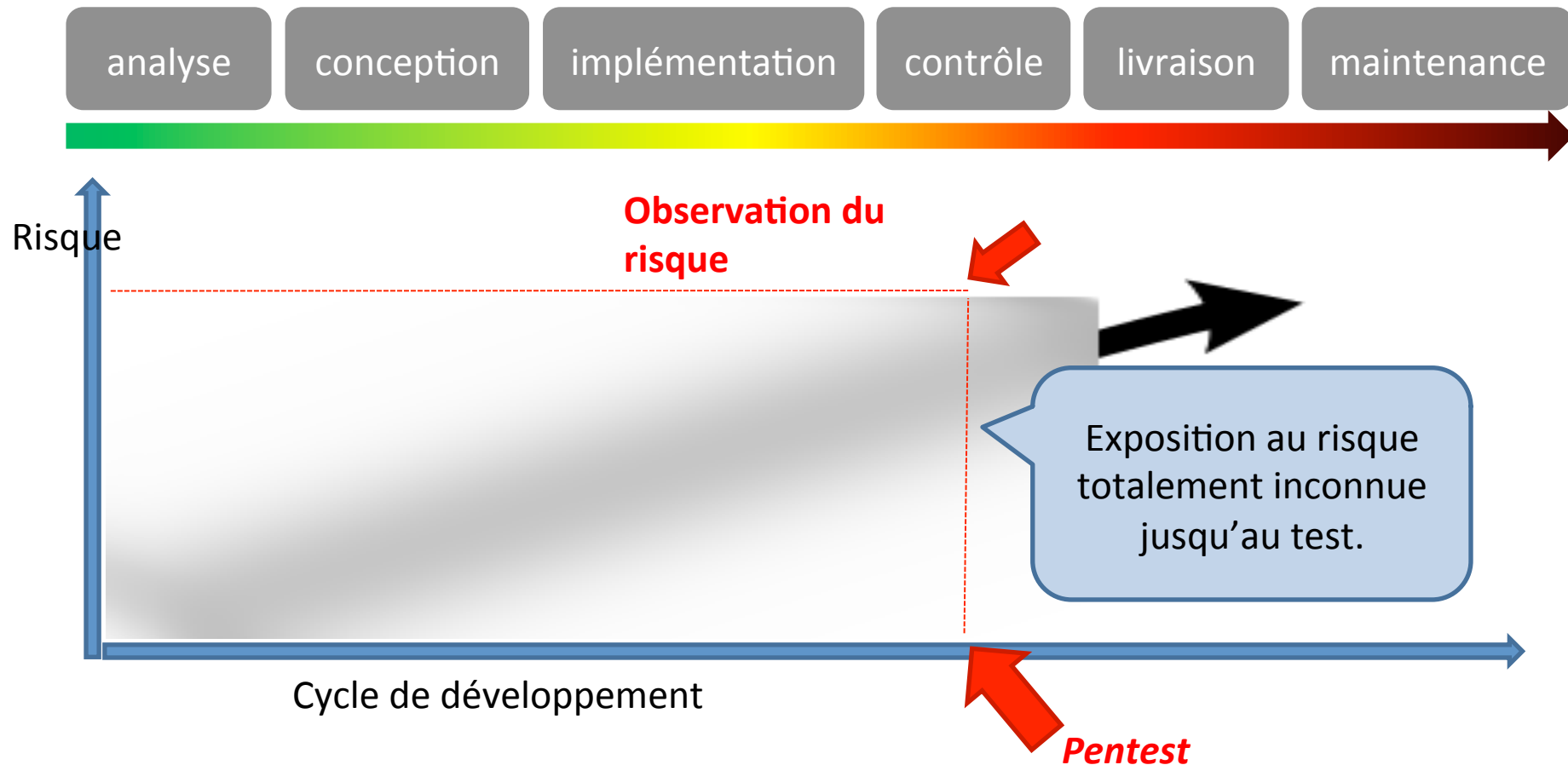
contrôle

livraison

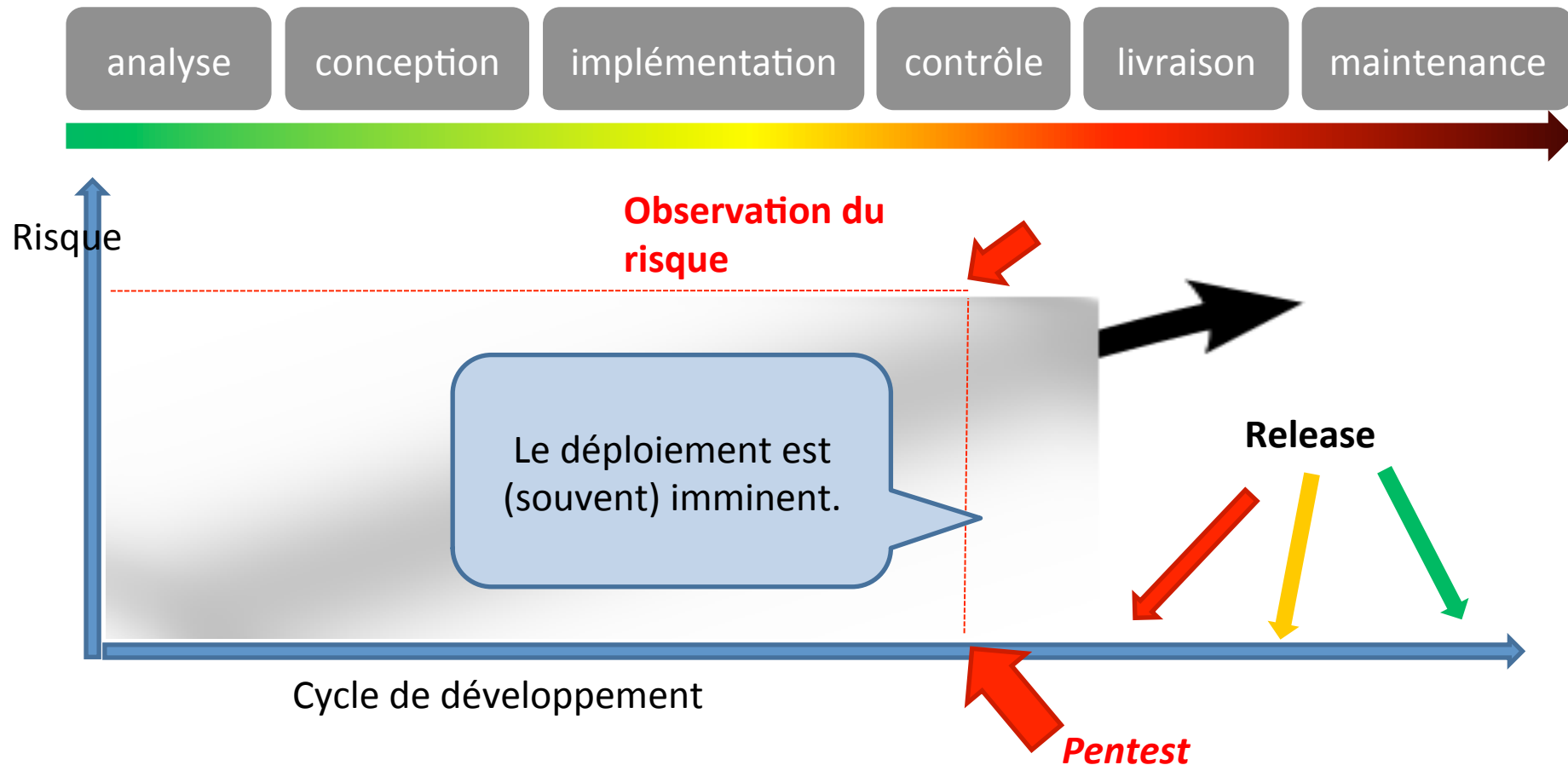
maintenance



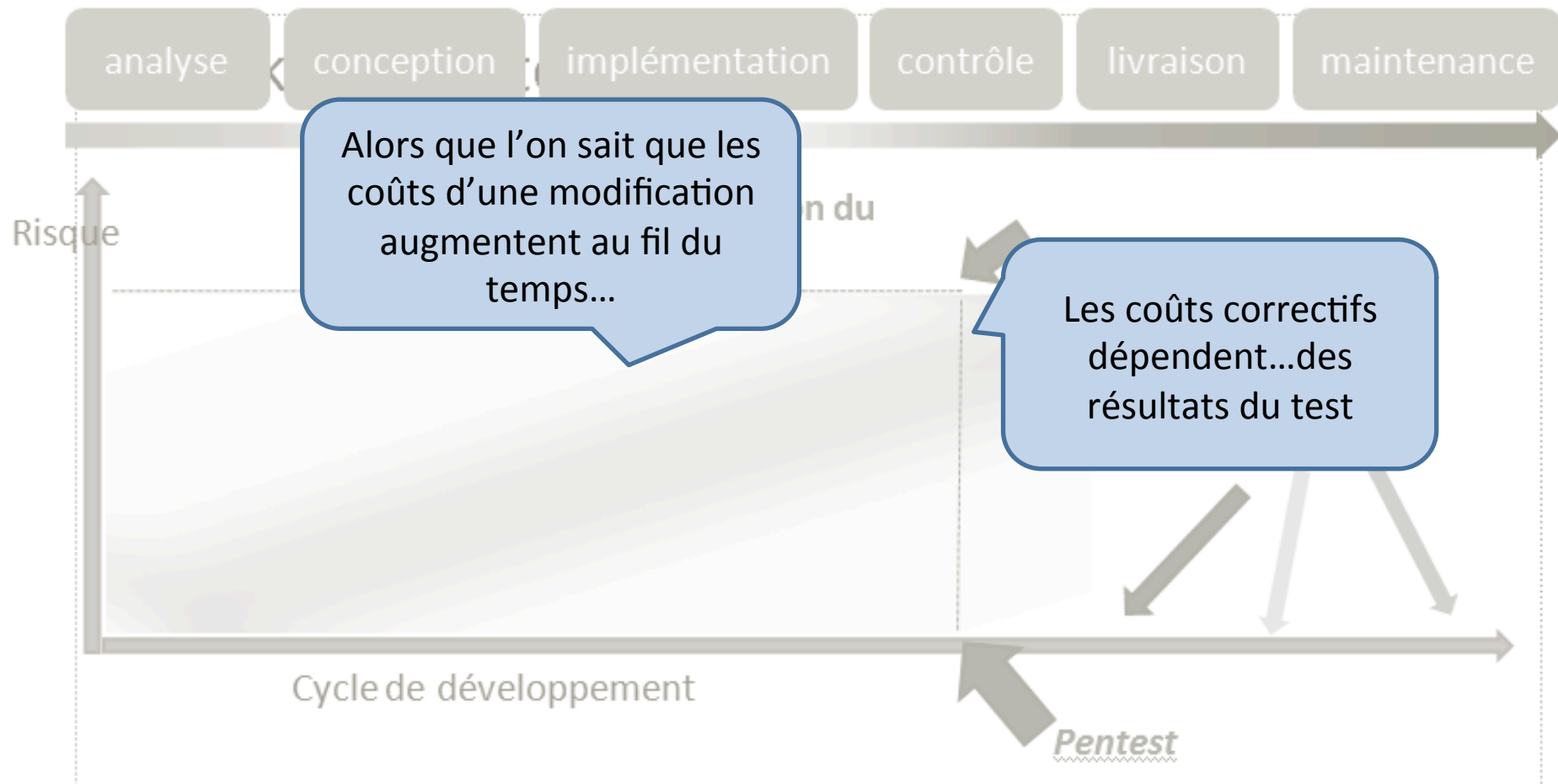
Le paradigme « pentest »



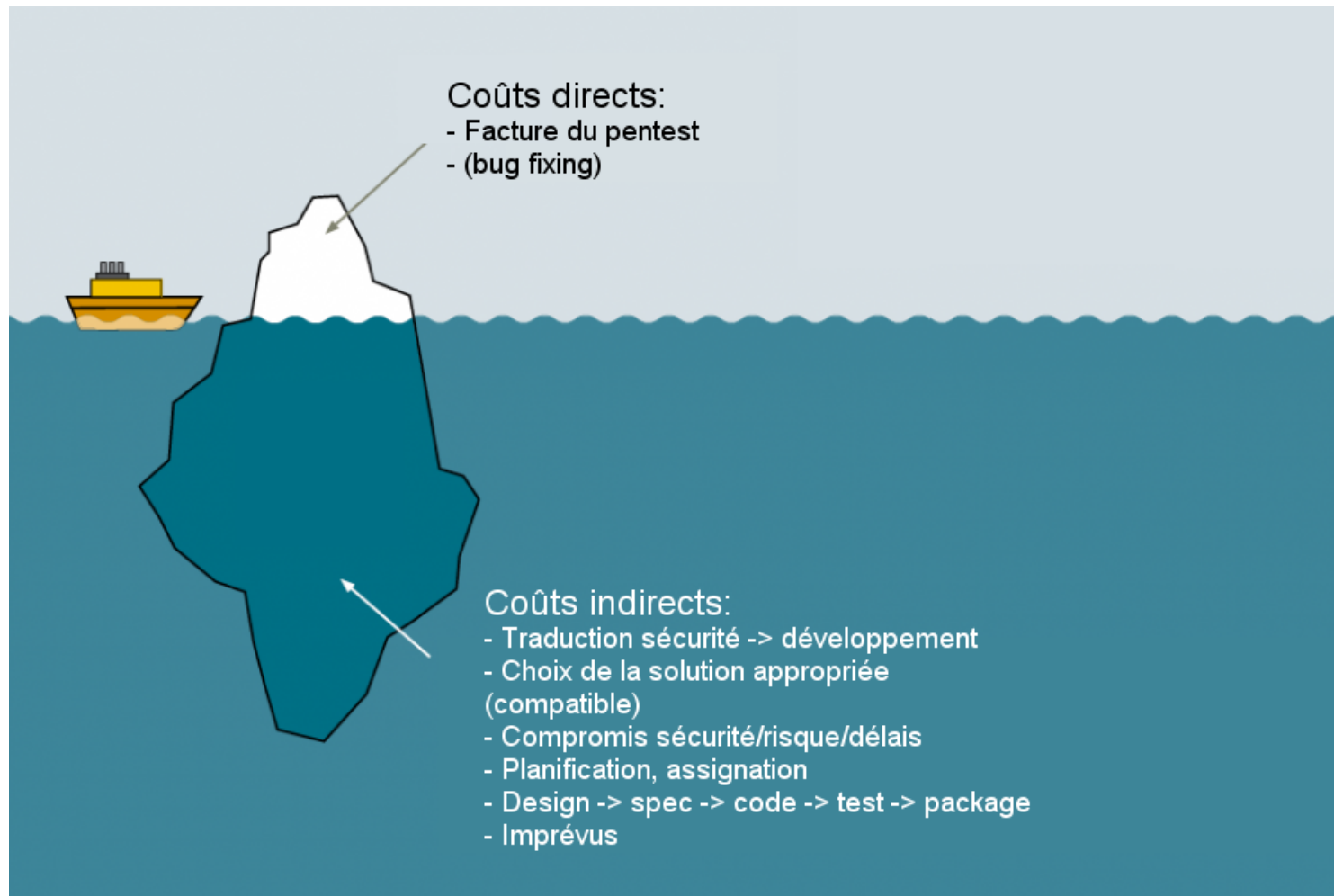
Le paradigme « pentest »



Le paradigme « pentest »



Le paradigme « pentest »

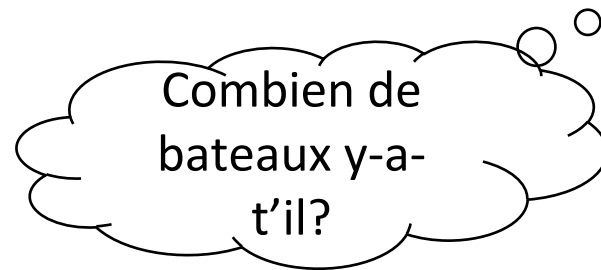


Le paradigme « pentest »

- Vision limitée du risque:
 - Par le scope
 - Par le délai
 - Par les connaissances du testeur



neilsingapore@flickr.com



Le paradigme « pentest »

- La réalité:
 - Plusieurs centaines d'applications en production
 - Un SDLC par équipe de développement
 - Des difficultés de communication
 - Des « urgences » et des « changements »
 - Etc.
- Trop de « surprises »
- Capitalisation difficile



Quel est le besoin « terrain »?

1. Identifier le risque

→ à chaque phase du projet de développement!

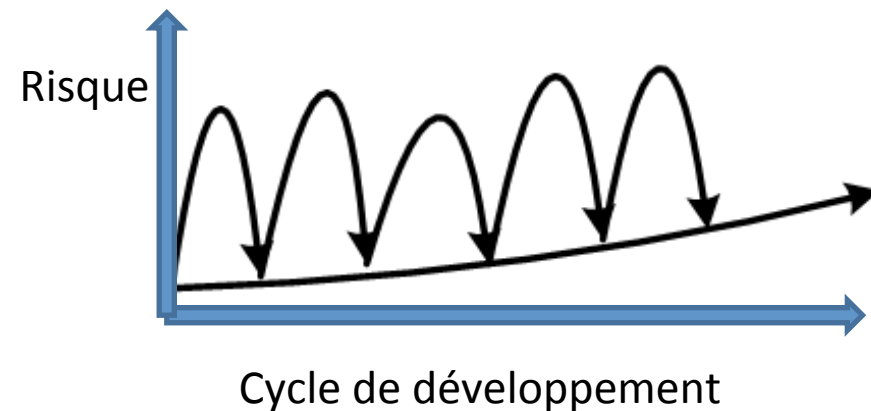
→ sur la base de l'état des connaissances

2. Gérer le risque

→ s'aligner sur les attentes/exigences de l'organisation

→ réduire le risque à
chaque fois que c'est
possible

→ prioriser les actions



Quel est le besoin « terrain »?

3. Optimiser les coûts de la sécurité:

→ *agir lorsque les coûts de correction ou détection sont faibles*

→ *capitaliser les erreurs*

4. Disposer d'une approche formelle:

→ *pouvant être répétée*

→ *adaptative: supportant des cycles (SDLC) hétérogènes*

→ *évolutive: prévue pour une adoption progressive*

→ *mesurable: la progression est formelle et compatible avec une approche qualité*



Quel est le besoin « terrain »?

5. Une boîte à outils:

- *une approche « toolbox » facilitant le développement inter-unités/organisations*
- *un mode d'emploi, accessible aux non-experts*
- *des supports prêts à l'emploi (approche prescriptive) et agnostiques*
- *une communauté*

➔ Modèle de maturité

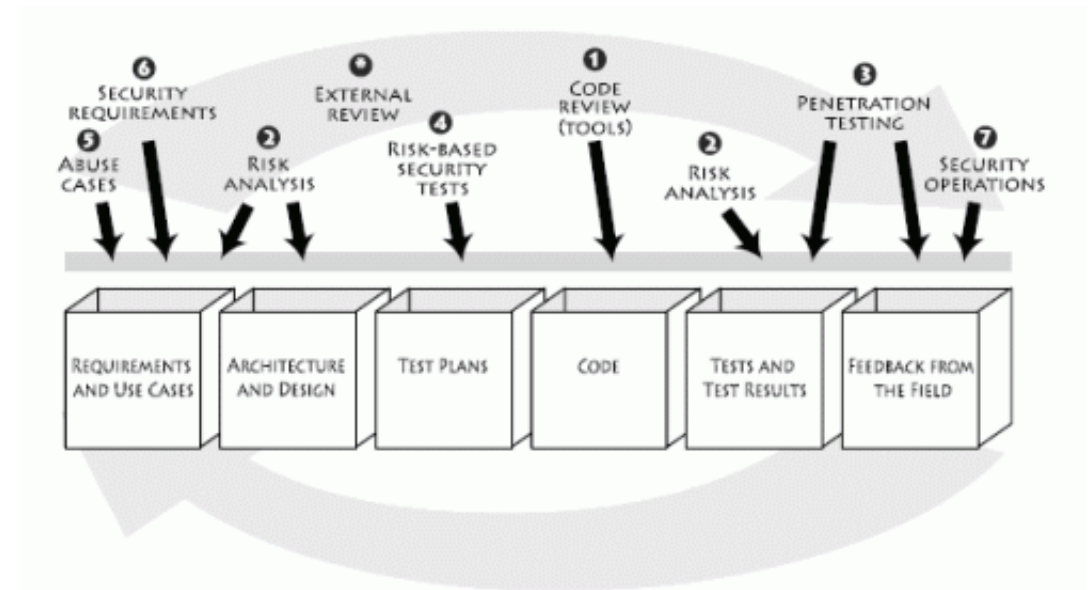
Les solutions alternatives...



Touchpoints



**SOFTWARE
SECURITY**
BUILDING SECURITY IN



- 2007, Conçu par *Cigital*
- 7 activités de sécurité
- La méthode n'est pas maintenue (mais son efficacité n'est pas remise en question!)

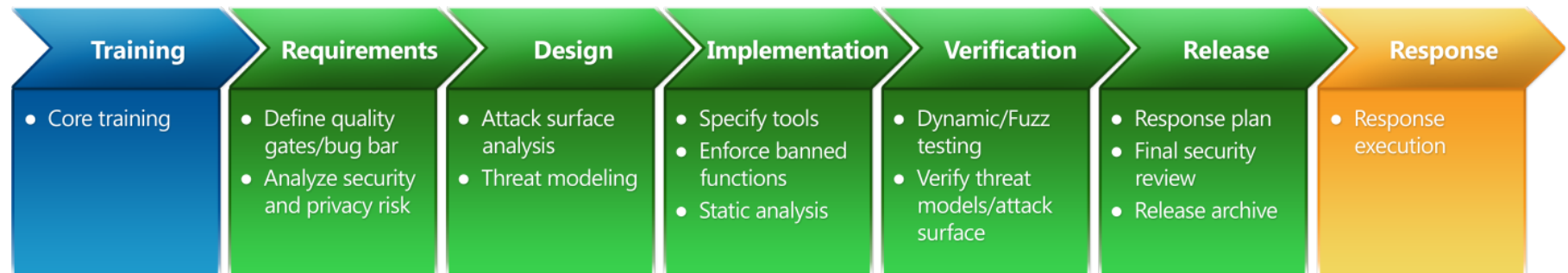
<http://www.swsec.com/>



SDL



Microsoft® Security Development Lifecycle



- 2010 (5 itérations), Conçu par *Microsoft*
- 14 activités de sécurité
- Approche “éditeur logiciel”
 - Modèle réduit pour “équipes agiles”



B-SIMM2



The Software Security Framework (SSF)			
Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

- 2010 (v.2), Conçu par *Cigital et Fortify*
- 12 activités de sécurité, 4 domaines
- Approche centrée sur l'observation:
 - Description des cycles en œuvre dans 30 organisations sécurisant leurs logiciels



OpenSAMM



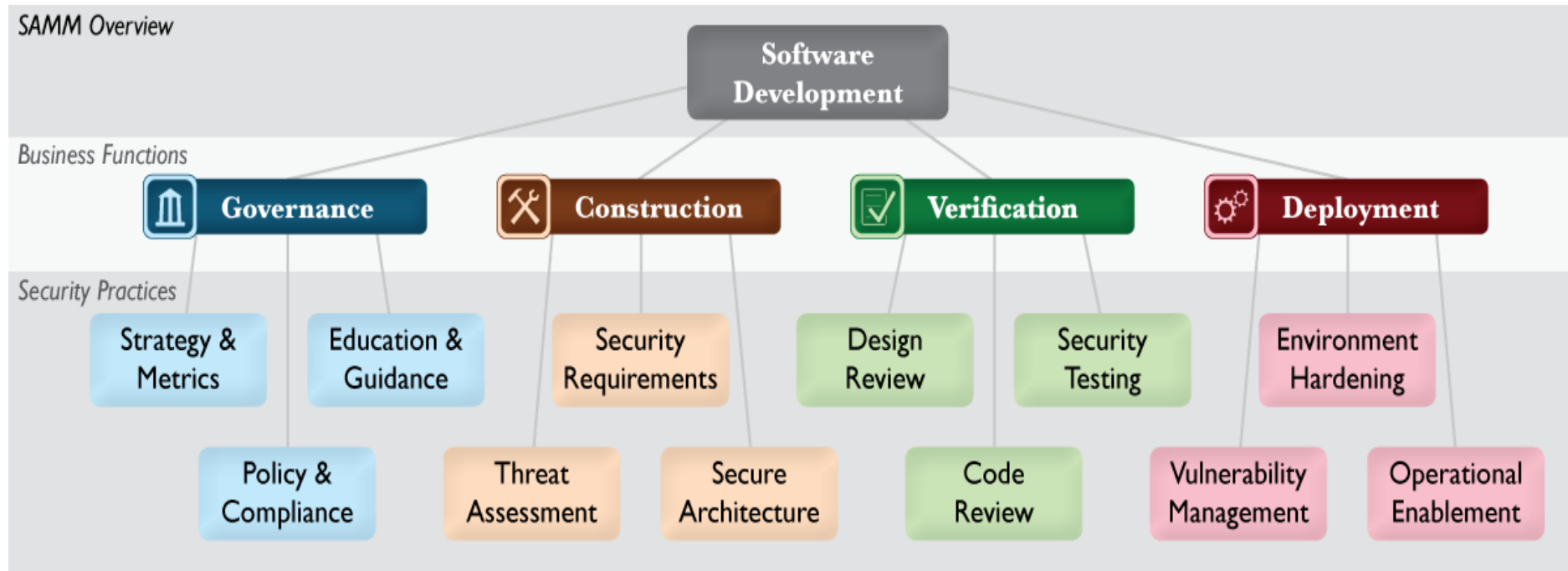
OpenSAMM



- Méthodologie conçue par **Pravir Chandra**, beta publiée en 2008, v.1. en 2009
sur: <http://www.opensamm.org>
- Ouvert, gratuit, non-exclusif, agnostique:
- Modèle de maturité, proposant:
 - 4 domaines d'activités (*disciplines*) regroupant au total 12 activités de sécurité (*security practices*)
 - 3 niveaux de maturité par domaine
 - Une approche « gouvernance »



OpenSAMM



OpenSAMM

Verification

Description of Security Practices



Design Review

The Design Review (DR) Practice is focused on assessment of software design and architecture for security-related problems. This allows an organization to detect architecture-level issues early in software development and thereby avoid potentially large costs from refactoring later due to security concerns.

Beginning with lightweight activities to build understanding of the security-relevant details about an architecture, an organization evolves toward more formal inspection methods that verify completeness in provision of security mechanisms. At the organization level, design review services are built and offered to stakeholders.

In a sophisticated form, provision of this Practice involves detailed, data-level inspection of designs and enforcement of baseline expectations for conducting design assessments and reviewing findings before releases are accepted.



Code Review

The Code Review (CR) Practice is focused on inspection of software at the source code level in order to find security vulnerabilities. Code-level vulnerabilities are generally simple to understand conceptually, but even informed developers can easily make mistakes that leave software open to potential compromise.

To begin, an organization uses lightweight checklists and for efficiency, only inspects the most critical software modules. However, as an organization evolves it uses automation technology to dramatically improve coverage and efficacy of code review activities.

Sophisticated provision of this Practice involves deeper integration of code review into the development process to enable project teams to find problems earlier. This also enables organizations to better audit and set expectations for code review findings before releases can be made.



Security Testing

The Security Testing (ST) Practice is focused on inspection of software in the runtime environment in order to find security problems. These testing activities bolster the assurance case for software by checking it in the same context in which it is expected to run, thus making visible operational misconfigurations or errors in business logic that are difficult to otherwise find.



OpenSAMM

Code Review

...more on page 62



OBJECTIVE

Opportunistically find basic code-level vulnerabilities and other high-risk security issues

Make code review during development more accurate and efficient through automation

Mandate comprehensive code review process to discover language-level and application-specific risks

ACTIVITIES




- A. Create review checklists from known security requirements
- B. Perform point-review of high-risk code

- A. Utilize automated code analysis tools
- B. Integrate code analysis into development process

- A. Customize code analysis for application-specific concerns
- B. Establish release gates for code review

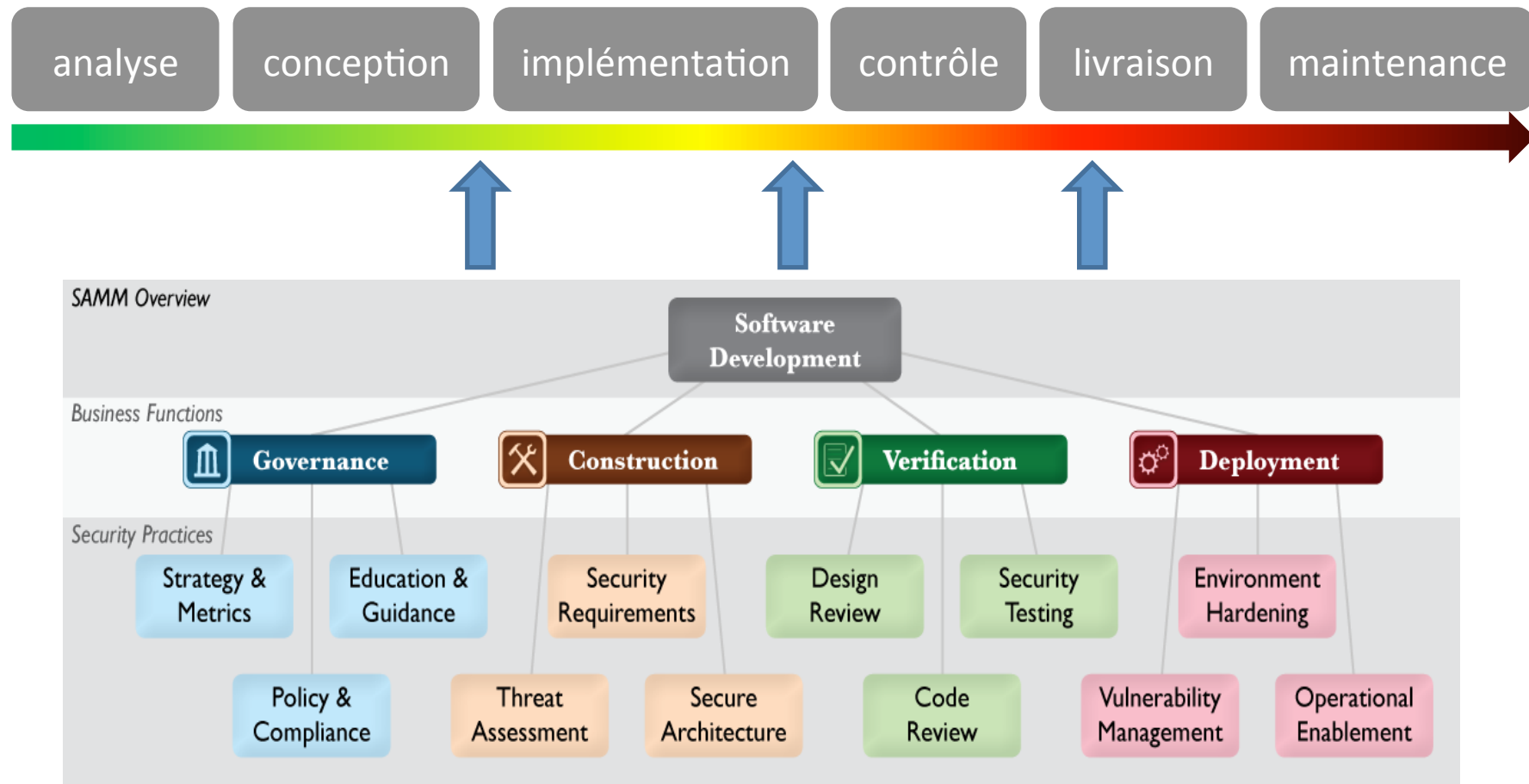


OpenSAMM

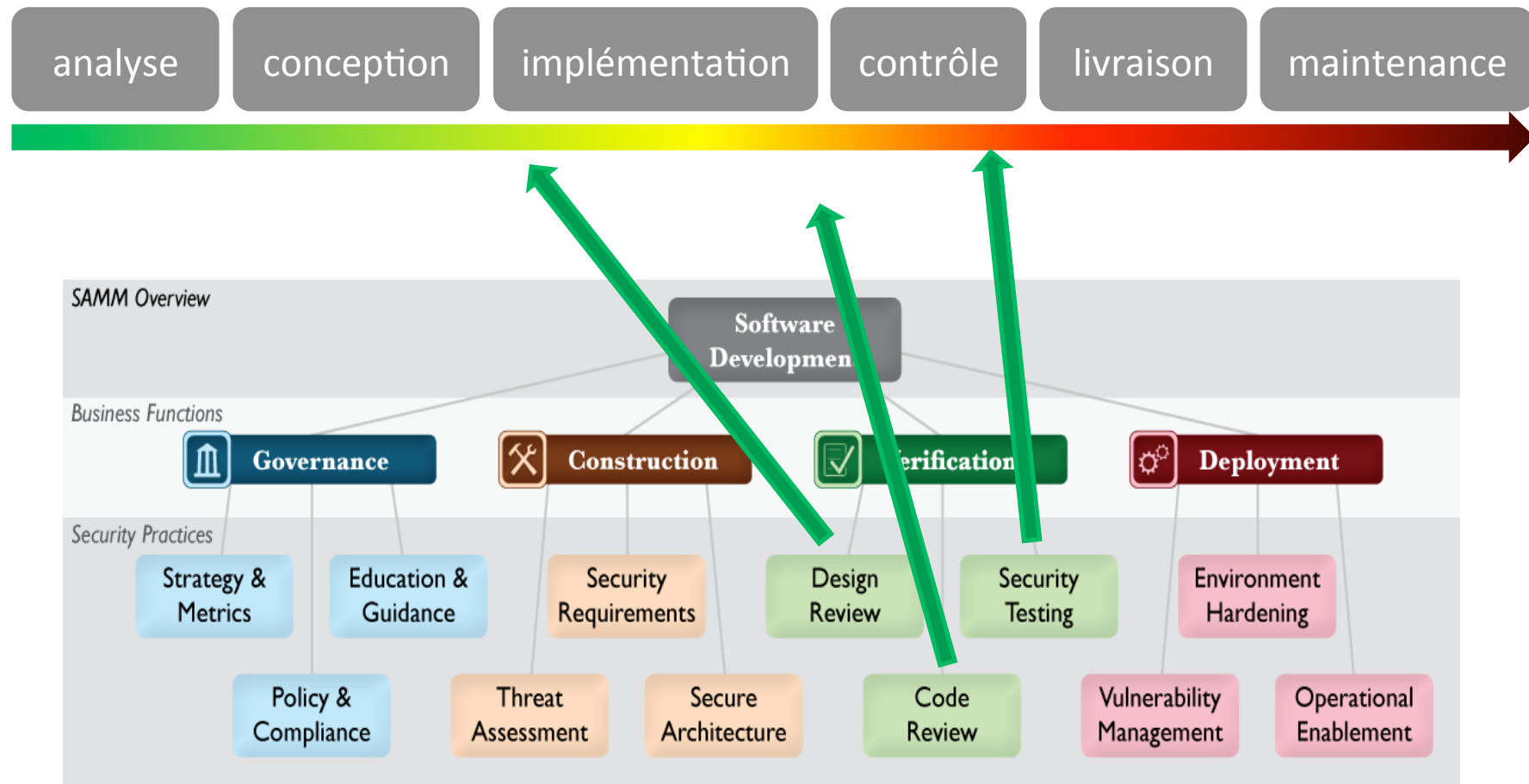
	 CR 1	 CR 2	 CR 3
OBJECTIVE	Opportunistically find basic code-level vulnerabilities and other high-risk security issues	Make code review during development more accurate and efficient through automation	Mandate comprehensive code review process to discover language-level and application-specific risks
ACTIVITIES	<p>A. Create review checklists from known security requirements</p> <p>B. Perform point-review of high-risk code</p>	<p>A. Utilize automated code analysis tools</p> <p>B. Integrate code analysis into development process</p>	<p>A. Customize code analysis for application-specific concerns</p> <p>B. Establish release gates for code review</p>
ASSESSMENT	<ul style="list-style-type: none"> ◆ Do most project teams have review checklists based on common problems? ◆ Are project teams generally performing review of selected high-risk code? 	<ul style="list-style-type: none"> ◆ Can most project teams access automated code analysis tools to find security problems? ◆ Do most stakeholders consistently require and review results from code reviews? 	<ul style="list-style-type: none"> ◆ Do project teams utilize automation to check code against application-specific coding standards? ◆ Does routine project audit require a baseline for code review results prior to release?
RESULTS	<ul style="list-style-type: none"> ◆ Inspection for common code vulnerabilities that lead to likely discovery or attack ◆ Lightweight review for coding errors that lead to severe security impact ◆ Basic code-level due diligence for security assurance 	<ul style="list-style-type: none"> ◆ Development enabled to consistently self-check for code-level security vulnerabilities ◆ Routine analysis results to compile historic data on per-team secure coding habits ◆ Stakeholders aware of unmitigated vulnerabilities to support better tradeoff analysis 	<ul style="list-style-type: none"> ◆ Increased confidence in accuracy and applicability of code analysis results ◆ Organization-wide baseline for secure coding expectations ◆ Project teams with an objective goal for judging code-level security



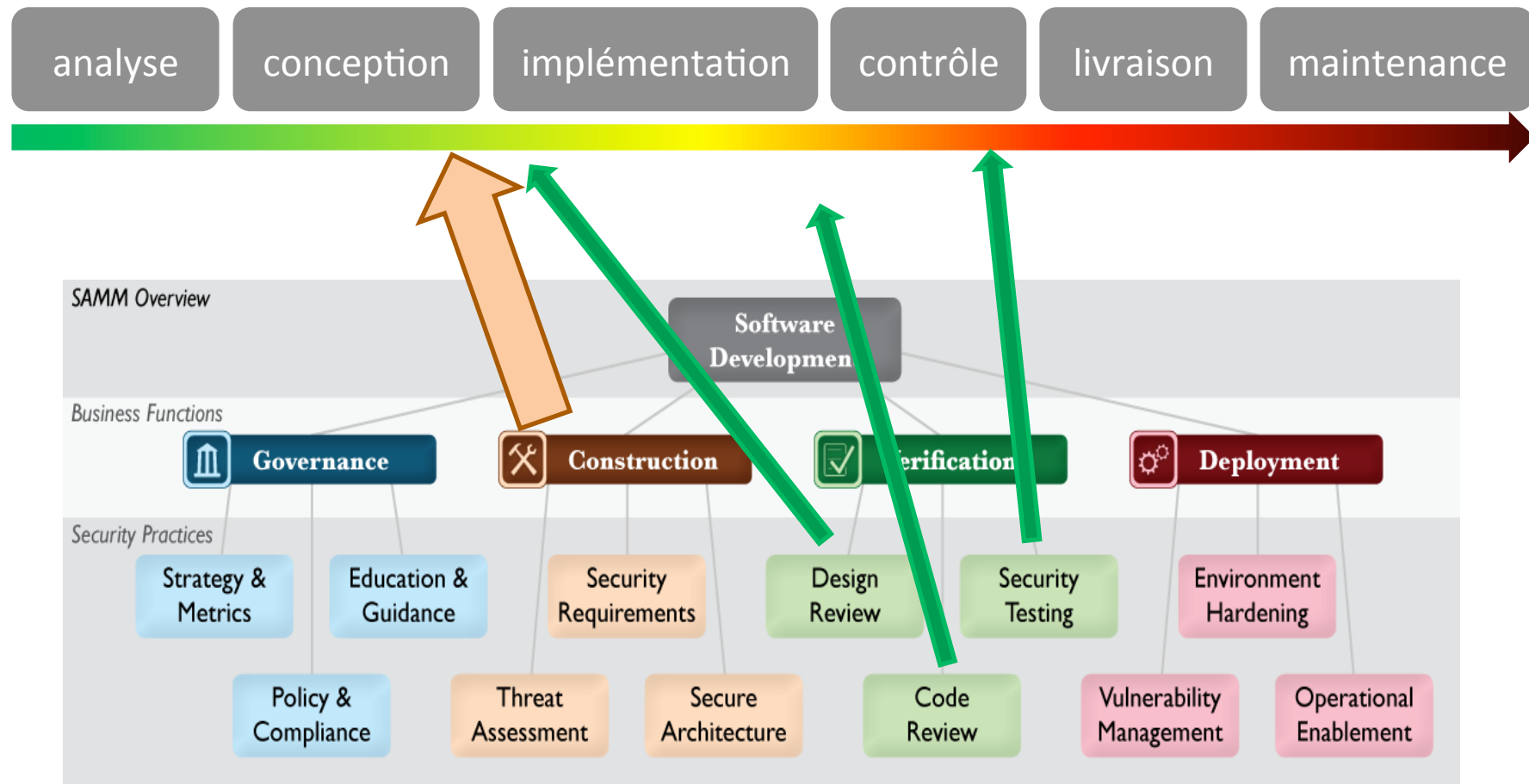
OpenSAMM



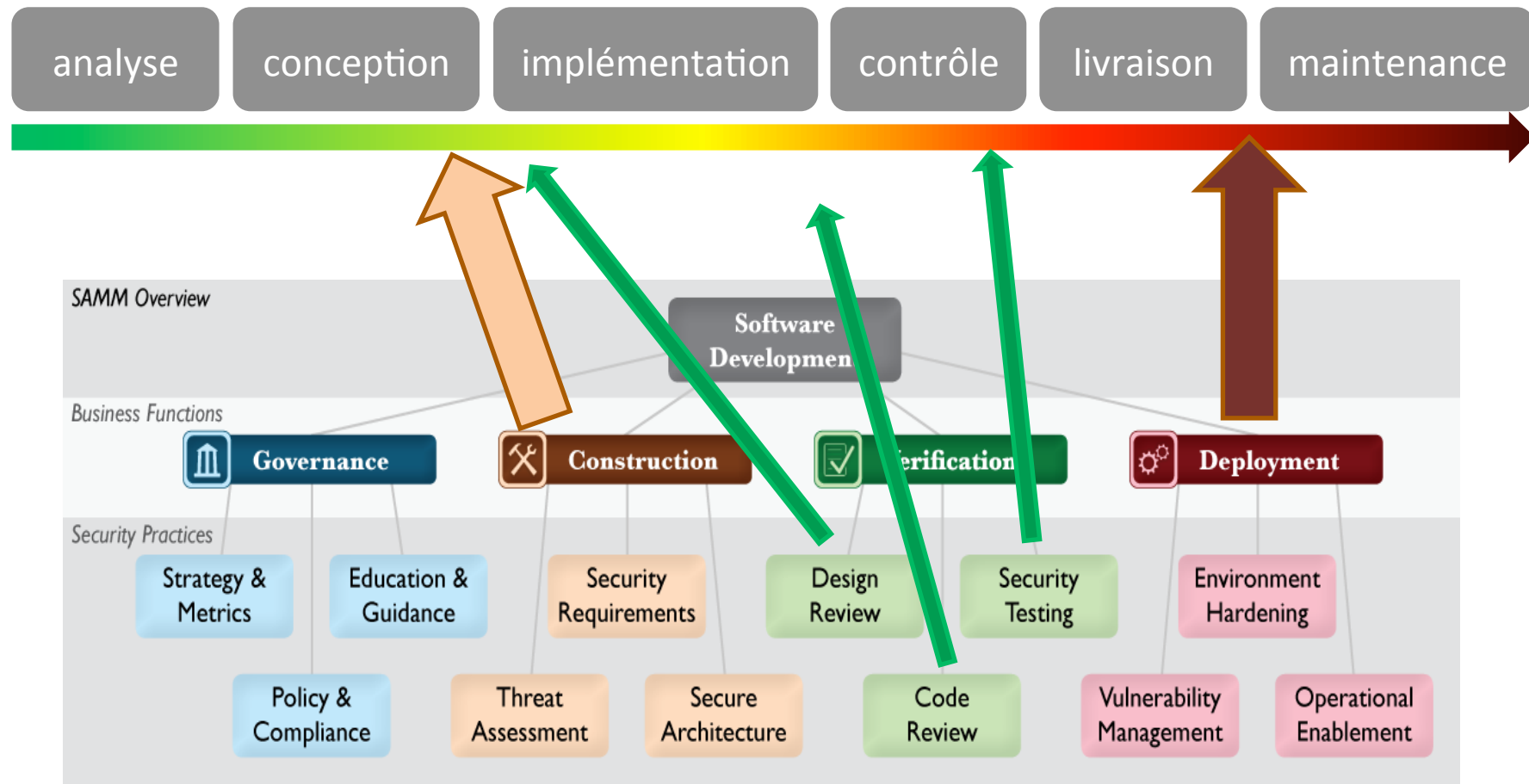
OpenSAMM



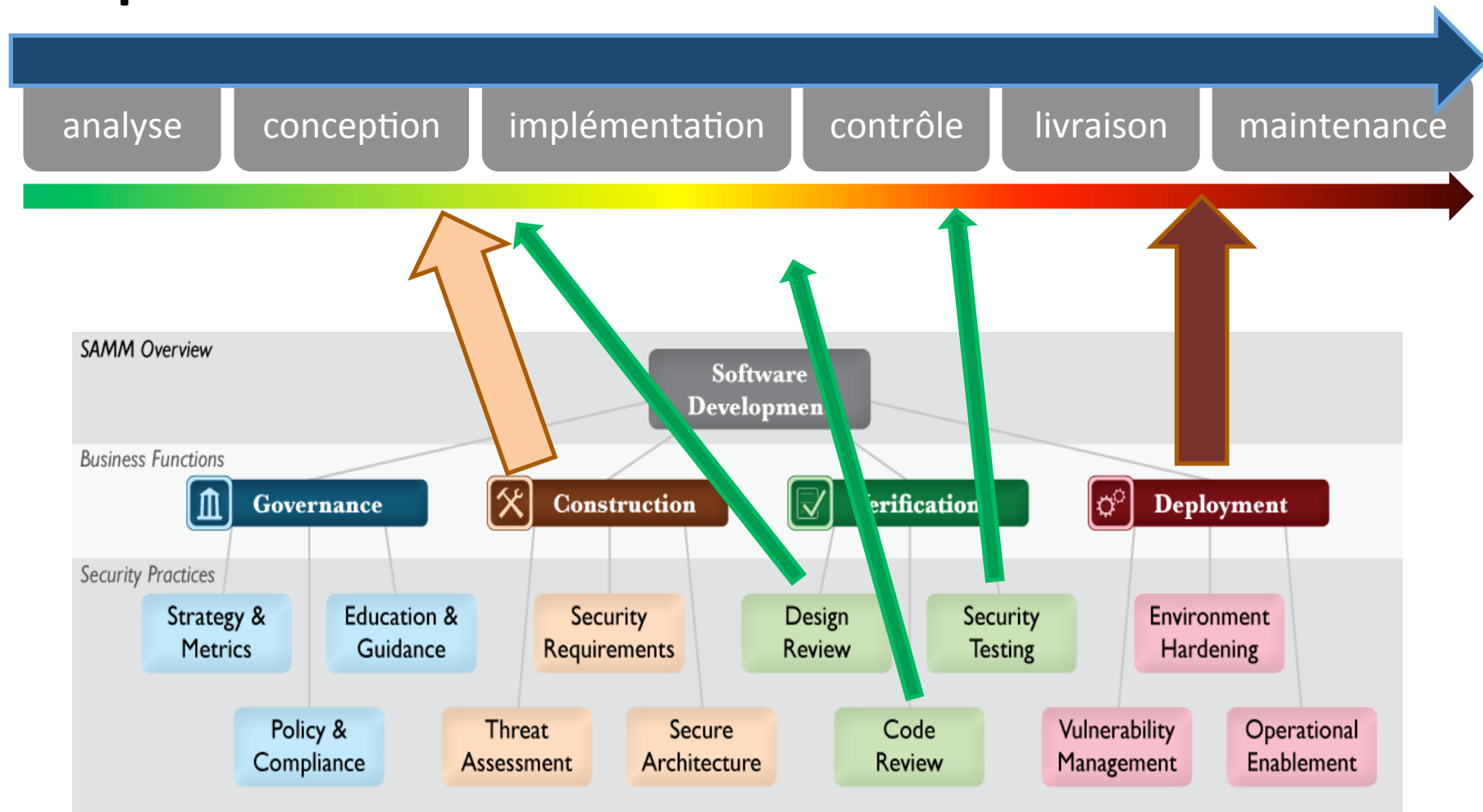
OpenSAMM



OpenSAMM



OpenSAMM



OpenSAMM

- Questionnaires pour l'évaluation du niveau de maturité (établissement des *scorecards*)

Code Review		Yes/No
◆ Do most project teams have review checklists based on common problems?		
◆ Are project teams generally performing review of selected high-risk code?		
◆ Can most project teams access automated code analysis tools to find security problems?		<input checked="" type="checkbox"/> CR 1
◆ Do most stakeholders consistently require and review results from code reviews?		<input checked="" type="checkbox"/> CR 2
◆ Do project teams utilize automation to check code against application-specific coding standards?		
◆ Does routine project audit require a baseline for code review results prior to release?		<input checked="" type="checkbox"/> CR 3



OpenSAMM

- Exemples de plans de maturité pour diverses industries:
 - Éditeur logiciel
 - Fournisseur de services en ligne
 - Institution financière
 - Secteur public



OpenSAMM: l'avenir

- Correspondance avec les standards et référentiels existants: COBIT, ISO 27002, PCI
- Ajout de nouveaux modèles de programmes de sécurité
- Ajout d'études de cas
- Nouvelle version
 - (cycle de 6-12 mois → à *prendre entre pincettes*)



Débuter avec OpenSAMM

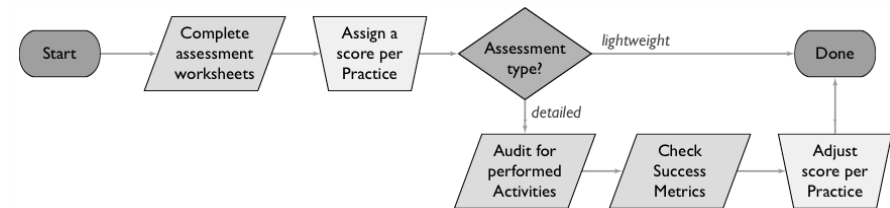


Premières actions:

- Qualifier le besoin:
 - Prendre la température
 - Démarrer ou aligner un programme de développement sécurisé
 - Faire paniquer la Direction



Premières actions:

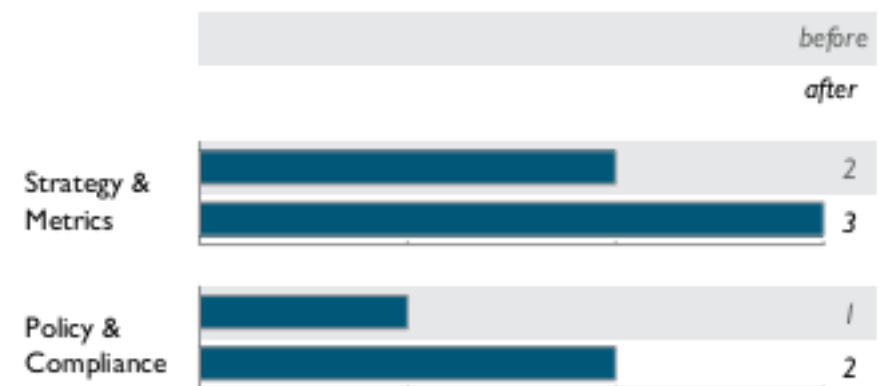


- Conduire l'évaluation (*assessment*)

- Utiliser les guides disponibles (feuilles Excel mises à disposition par la communauté)
- Profondeur proportionnelle au besoin identifié
- Construire les *scorecards*:

Conduire l'évaluation même si l'on pense qu'il n'y a rien!

→ Bcp d'équipes ne formalisent pas leurs activités de sécurité!

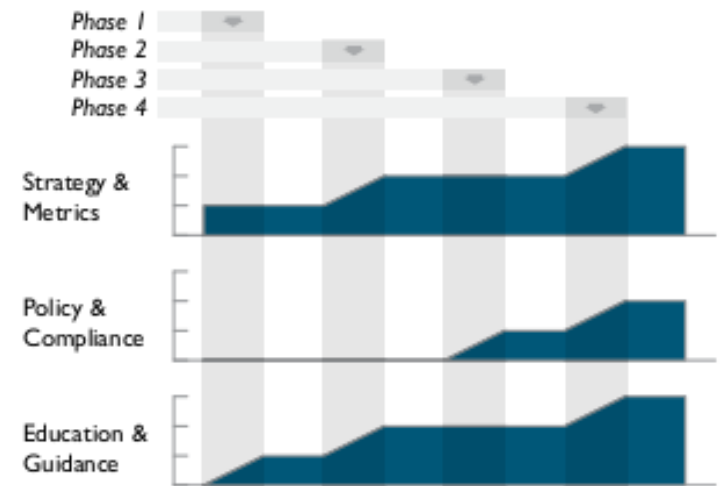


Premières actions:

- Identifier l'opportunité:

- Démarrer un programme de sécurité?
- Formaliser/renforcer le programme existant ?
- Déléguer / accompagner la mise en œuvre?

Privilégier l'itération à la planification (éviter la « *roadmap magique* »)!



Et ensuite?

- Itération:

- Ajouter une activité?

- Accroître le niveau de maturité d'une activité existante?



A éviter...



Pièges

1. Viser l'implémentation totale d'OpenSAMM

- OpenSAMM est un modèle de maturité, pas une méthodologie!
- Conserver le focus sur la réduction de risques et les opportunités présentes au sein de l'organisation (compétences, ressources)
- Ne pas privilégier l'implémentation à 100% mais renforcer les activités à haute valeur ajoutée!



Pièges

2. Confondre « niveau de maturité » avec « maîtrise de l'activité »

– Que l'on soit niveau 1, 2 ou 3 ne fournit aucune indication quant au degré de maîtrise de l'activité (« prouver que l'on fait » vs. « savoir faire correctement »)

– P.ex:

Code Review	Yes/No
◆ Do most project teams have review checklists based on common problems?	
◆ Are project teams generally performing review of selected high-risk code?	<input checked="" type="checkbox"/> CR 1
◆ Can most project teams access automated code analysis tools to find security problems?	
◆ Do most stakeholders consistently require and review results from code reviews?	<input checked="" type="checkbox"/> CR 2
◆ Do project teams utilize automation to check code against application-specific coding standards?	
◆ Does routine project audit require a baseline for code review results prior to release?	<input checked="" type="checkbox"/> CR 3



Pièges

3. Confondre confiance et assurance

- OpenSAMM est un modèle de confiance et non d'assurance.
- Zones caractéristique de fuite:
 - Contraintes réglementaires incomprises
 - Formateurs hors sujet ou peu pédagogues
 - Architectures/librairies considérées comme sûres alors qu'elles sont vulnérables
 - Tests insuffisants dans les activités de contrôle
 - Écarts/divergences entre le code testé et le code déployé
 - Etc.



Pièges

En voyez-vous d'autres?



Conclusion

- Modèle de maturité « clés en mains » à la libre disposition des organisations
- Si implémenté correctement, accroît la visibilité sur les risques
- Apporte un soutien à des directives normées ou standardisées du type « mettre en œuvre une stratégie de développement sécurisé »
- Vision prescriptive de l'organisation générale du programme



Conclusion

- Vision évolutive, adaptable aux cycles existants et technologiquement agnostique
- Un modèle supporté par la communauté, appuyé par des outils
- Non-exclusif, interopérable
- Toutefois:
 - N'apporte aucune réponse technique
 - La maturité du modèle de maturité lui-même est encore à identifier...



Questions?



Merci!

Pour me contacter:

antonio.fontes@owasp.org
@starbuck3000

<http://www.opensamm.org>

