

## Guide Complet Scrum Master : PSM II - Niveau Avancé

### Professional Scrum Master II - Mastery Level

---

"PSM II n'est pas sur CE QUE vous savez, mais sur COMMENT vous l'appliquez dans le chaos réel"

---

### Table des Matières - Partie 2 (PSM II)

1. Comprendre PSM II : Ce Qui Change
  2. Servant Leadership Mastery
  3. Coaching Professionnel : Frameworks et Pratiques
  4. Facilitation Avancée : Au-delà du Basique
  5. Coaching du Product Owner
  6. Coaching des Developers
  7. Transformation Organisationnelle
  8. Evidence-Based Management (EBM)
  9. Scaling Scrum : Nexus Framework
  10. Métriques et Flow
  11. Situations Complexes et Anti-Patterns
  12. Préparation Exam PSM II
  13. 100 Questions Scénarios PSM II
- 

## 1. Comprendre PSM II : Ce Qui Change

### 1.1 PSM I vs PSM II : La Vraie Différence

#### Comparaison Détaillée

Aspect	PSM I	PSM II
Focus	Connaissance théorique	Application pratique
Questions	"Qu'est-ce que X ?"	"Comment gérer Y ?"

Aspect	PSM I	PSM II
<b>Scrum Guide</b>	Connaissance exacte	Application nuancée
<b>Expérience</b>	Pas requise	Fortement recommandée (1+ an)
<b>Difficulté</b>	Modérée	Élevée
<b>Type questions</b>	Directes, QCM	Scénarios longs, contextuels
<b>Temps réflexion</b>	45s/question	2-3 min/question
<b>Taux réussite</b>	~50%	~30%
<b>Ce qu'on teste</b>	Mémoire + compréhension	Jugement + expérience


---

### Exemple Concret de Différence

**Même Sujet : Product Owner absent**

**Question Type PSM I :**

Q: Qui est responsable de la disponibilité du Product Owner pour l'équipe ?

- A. Scrum Master
- B. Product Owner lui-même 
- C. Management
- D. Toute la Scrum Team

→ Réponse factuelle, dans le Scrum Guide

**Question Type PSM II :**

Scénario :

Le Product Owner de votre équipe est le VP Product de l'entreprise. Il assiste au Sprint Planning et Sprint Review, mais n'est jamais disponible pendant le Sprint. Les Developers doivent attendre 3-5 jours pour des clarifications sur les

stories, causant des blocages constants. Plusieurs stories sont livrées mais rejetées en Review car "pas ce que je voulais".

La vélocité chute de 35 à 15 points. Les Developers sont frustrés. Le PO dit "Je suis très occupé, ils doivent se débrouiller".

Vous avez déjà discuté avec le PO en 1-on-1 mais rien n'a changé.

Question :

Quelle devrait être votre PROCHAINE action en tant que Scrum Master ?

- A. Escalader au CEO que le PO ne remplit pas son rôle
- B. Former un Developer pour jouer rôle de "proxy PO"
- C. Faciliter une réunion avec PO, son manager et l'équipe pour discuter impact et trouver solutions durables
- D. Accepter la situation et coacher l'équipe à être plus autonome sur les décisions produit
- E. Proposer de réduire la durée des Sprints pour avoir plus de touchpoints avec le PO

→ Teste jugement, expérience, nuances

→ Plusieurs réponses "raisonnables", une seule optimale

→ Nécessite comprendre dynamiques organisationnelles

### **Analyse de la Question PSM II :**

Pourquoi A est mauvais :

- Trop agressif, casse relations
- Pas encore épuisé toutes options

- SM doit résoudre à son niveau d'abord

Pourquoi B est mauvais :

- Viole Scrum (pas de proxy PO)
- Ne résout pas le vrai problème
- PO reste accountable

Pourquoi C est correct : 

- Escalation appropriée (le manager du PO)
- Facilitation neutre, pas accusation
- Cherche solution systémique
- Implique parties prenantes clés
- SM joue son rôle de servant leader

Pourquoi D est mauvais :

- Abandonne trop vite
- PO est essentiel, on ne peut pas juste "accepter"
- Autonomie ≠ remplacer le PO

Pourquoi E est sub-optimal :

- Traite symptôme, pas cause
- Sprints plus courts = plus de cérémonie = moins de temps
- Ne garantit pas disponibilité PO

**Voyez la différence ? **







- PSM I = Connaissance pure
- PSM II = Jugement dans le chaos

---




## 1.2 Format de l'Examen PSM II

## Détails Techniques

### Structure :

-  **30 questions** (vs 80 pour PSM I)
-  **90 minutes** (3 min/question moyenne)
-  **85% minimum** (26/30 bonnes réponses)
-  **\$250 USD**
-  **Online**
-  **Marge d'erreur : 4 questions seulement !**

### Types de Questions :

-  **Scénarios longs** (80% des questions)
  - 2-5 paragraphes de contexte
  - Situations réelles complexes
  - Multiples layers de problèmes
-  **Multiple select** (plusieurs bonnes réponses possibles)
  - "Sélectionnez 3 actions appropriées..."
  - Toutes doivent être correctes
-  **Questions de jugement**
  - "Quelle est la MEILLEURE approche ?"
  - "Que devriez-vous faire EN PREMIER ?"
  - "Quelle serait votre action la PLUS EFFICACE ?"

### Pièges Spécifiques PSM II :

#### 1. Réponses "toutes bonnes" mais une meilleure

Scénario : Conflit dans l'équipe

Toutes ces réponses sont "bonnes" :

A. Faciliter une discussion d'équipe

B. Parler individuellement avec chacun

C. Utiliser une rétrospective dédiée

D. Former l'équipe à la communication non-violente

Mais UNE est la MEILLEURE selon le contexte spécifique

→ Faut analyser nuances du scénario

## **2. Tentations "quick fix"**

Réponses "rapides" sont souvent mauvaises :

- "Dire à X de faire Y"
- "Prendre la décision soi-même"
- "Escalader immédiatement"

Réponses correctes PSM II :

- Faciliter, coacher, guider
- Faire émerger solutions
- Développer autonomie

## **3. Scénarios multicouches**

Un scénario PSM II inclut souvent :

- Problème d'équipe
- + Problème organisationnel
- + Problème de PO
- + Problème de qualité
- + Pression management

Question : Quelle est votre PRIORITÉ ?

→ Teste capacité à trier et prioriser

---

### **1.3 Compétences Testées PSM II**

 **Les 8 Domaines de Compétence**

## **1 Servant Leadership (25%)**

- Influencer sans autorité
- Guider vs diriger
- Développer autonomie équipe
- Leadership situationnel
- Modèle SCARF (neurosciences)

## **2 Coaching (25%)**

- Coaching d'équipe
- Coaching Product Owner
- Coaching organisation
- Poser questions puissantes
- Écoute active niveau 3
- Modèle GROW
- Coaching vs Mentoring vs Teaching

## **3 Facilitation (15%)**

- Techniques avancées (Liberating Structures)
- Gestion conflits
- Prise de décision collaborative
- Design de workshops
- Facilitation remote

## **4 Change Management (10%)**

- Transformation organisationnelle
- Résistance au changement
- Culture agile
- Coaching exécutif
- Patterns et anti-patterns organisationnels

## **5 Product Value (10%)**

- Aider PO maximiser valeur

- Evidence-Based Management
- Métriques au-delà de vélocité
- Product Goal et vision

### **6 Scaling (5%)**

- Nexus Framework
- Coordination multi-équipes
- Dépendances
- Integration

### **7 Professionalisme (5%)**

- Éthique
- Apprentissage continu
- Amélioration personnelle
- Communauté de pratique

### **8 Situations Complexes (5%)**

- Jugement contextuel
- Priorisation problèmes
- Décisions difficiles
- Trade-offs

---

## **2. Servant Leadership Mastery**

### **2.1 Les 10 Caractéristiques du Servant Leader**

**Créées par Robert Greenleaf (1970)**

Le Scrum Master EST un servant leader. Mais qu'est-ce que ça veut dire VRAIMENT ?

---

#### **Caractéristique 1 : LISTENING (Écoute) 🗣️**

**Définition :** Le servant leader écoute d'abord, cherche à comprendre avant d'être compris.

**Pourquoi c'est crucial pour SM :**

- Comprendre vrais problèmes (pas juste symptômes)
- Gagner confiance de l'équipe
- Identifier impediments cachés
- Faire sentir l'équipe valorisée

### **Les 3 Niveaux d'Écoute :**

#### Niveau 1 : Écoute Interne 🙊

"Je pense à ma réponse pendant que tu parles"

- Focalisé sur soi
- Planifie ce qu'on va dire
- Rate 70% de l'info

❌ Inefficace

#### Niveau 2 : Écoute Focalisée 😊

"J'écoute tes mots et je comprends"

- Attention sur l'autre
- Comprend le contenu
- Pose questions de clarification

✅ Bien, mais pas suffisant pour coaching

#### Niveau 3 : Écoute Globale 🙋

"J'écoute ce qui est dit ET non-dit"

- Attention totale
- Langage corporel
- Émotions sous-jacentes
- Silences significatifs
- Pattern dans plusieurs conversations

✅✅ Optimal pour Scrum Master

### **Pratique Concrète :**

Situation : Developer dit "Je suis bloqué sur cette story"

Niveau 1 (Mauvais) :

SM pense : "Ok il a besoin d'aide technique,

je vais lui donner la solution..."

→ N'écoute pas vraiment

Niveau 2 (Moyen) :

SM : "Qu'est-ce qui te bloque techniquement ?"

Developer : "L'API ne répond pas comme attendu"

SM : "As-tu regardé les logs ?"

→ Écoute les mots, résout problème technique

Niveau 3 (Excellent) :

SM observe : Ton fatigué, hésitation, évite contact visuel

SM : "Je sens qu'il y a peut-être plus que le problème technique..."

Developer : [pause] "Honnêtement, je ne suis pas sûr de

l'approche qu'on a choisie mais j'avais peur de le dire"

SM : "Dis-m'en plus..."

→ Découvre le VRAI problème (pas technique, mais peur de s'exprimer)

### **Questions d'Écoute Active :**


- ✓ "Qu'est-ce que tu ressens par rapport à ça ?"
- ✓ "Si je comprends bien, tu dis que... ?"
- ✓ "Qu'est-ce qui est le plus important pour toi ici ?"
- ✓ "J'ai remarqué [langage corporel], veux-tu en parler ?"
- ✓ [Silence] → Laisser l'autre remplir le silence

### **Exercice Pratique PSM II :**

Scénario exam :

"Pendant le Daily Scrum, un Developer dit rapidement 'Tout va bien' mais croise les bras et évite le regard de l'équipe. Vous remarquez que c'est le 3ème jour consécutif qu'il fait ça."

Question : Quelle devrait être votre action ?

- A. Ne rien faire, il a dit que tout va bien
- B. Lui demander devant l'équipe s'il y a un problème
- C. Après le Daily, lui parler en privé pour comprendre ce qui se passe 
- D. Escalader au PO qu'il y a un problème

Réponse : C

Pourquoi : Écoute niveau 3 = observer non-verbal,  
créer safe space pour discussion privée

---

### **Caractéristique 2 : EMPATHY (Empathie)**

**Définition :** Comprendre et partager les sentiments des autres. Se mettre à leur place.

#### **Empathie ≠ Sympathie**

SYMPATHIE : "Je suis désolé que tu te sentes mal"

→ Rester dans sa perspective, avoir pitié

EMPATHIE : "Je comprends pourquoi tu te sens comme ça,  
à ta place je ressentirais probablement la même chose"

→ Voir depuis leur perspective, comprendre sans juger

#### **Application pour SM :**

#### **Exemple 1 : Developer Résiste Scrum**

✗ Sans empathie :

"Tu dois suivre Scrum, c'est comme ça"

✓ Avec empathie :

"Je comprends que ces meetings quotidiens te semblent une perte de temps, surtout quand tu es dans le flow.

Tu as passé 10 ans à travailler solo et c'était efficace.

Le changement est difficile. Puis-je te montrer ce que d'autres ont découvert après avoir essayé ?"

→ Reconnaît ses sentiments, valide son expérience,  
puis guide doucement

### **Exemple 2 : PO Sous Pression**

✗ Sans empathie :

"Tu dois être disponible pour l'équipe, c'est ton rôle"

✓ Avec empathie :

"Je vois que tu as 5 stakeholders qui te demandent des choses contradictoires, ton CEO qui veut des résultats hier, et l'équipe qui a besoin de clarté.

C'est une position très difficile. Comment puis-je t'aider à gérer ces demandes ?"

→ Montre compréhension de la complexité de sa situation

### **Le Modèle SCARF (Neurosciences) 🧠**

Développé par David Rock, explique ce qui active menace/récompense dans le cerveau :

S - STATUS (Statut social)

C - CERTAINTY (Certitude)

A - AUTONOMY (Autonomie)

R - RELATEDNESS(Appartenance)

F - FAIRNESS (Justice)

**Application empathique :**

Situation : Équipe résiste à nouveau processus

Analyse SCARF du SM empathique :

S - STATUS : "Ils ont peur de perdre leur statut d'experts  
si on change leur façon de travailler"

C - CERTAINTY : "Ils ne savent pas à quoi s'attendre,  
l'incertitude stresse"

A - AUTONOMY : "Ils sentent qu'on leur impose quelque chose,  
perte de contrôle"

R - RELATEDNESS : "Ils ont peur que leur équipe close-knit  
soit perturbée"

F - FAIRNESS : "Pourquoi notre équipe doit changer mais pas  
les autres ?"

Action SM empathique :

→ Adresse chaque dimension

→ Crée sécurité psychologique

→ Implique équipe dans décisions (Autonomie)

→ Explique le "pourquoi" (Fairness)

→ Rassure sur appartenance (Relatedness)

### Questions Empathiques :

- ✓ "Comment te sens-tu par rapport à ce changement ?"
  - ✓ "Qu'est-ce qui serait le plus difficile pour toi ?"
  - ✓ "Si tu étais à ma place, que ferais-tu ?"
  - ✓ "Qu'est-ce qui te préoccupe le plus ?"
  - ✓ "Comment puis-je t'aider ?"
- 

### Caractéristique 3 : HEALING (Guérison) 🌱

**Définition :** Aider les gens à surmonter leurs difficultés émotionnelles et professionnelles.

#### Pour le SM :

- Créer environnement psychologiquement sûr
- Aider équipe surmonter échecs
- Transformer conflits en apprentissage
- Guérir culture toxique

### Concept : Psychological Safety (Amy Edmondson) 🧠

Psychological Safety =

"Je peux prendre des risques interpersonnels sans peur de conséquences négatives pour mon image, statut ou carrière"

Caractéristiques équipe "safe" :

- ✓ On peut admettre erreurs sans punition
- ✓ On peut poser questions "bêtes"
- ✓ On peut challenger idées (même du senior)
- ✓ On peut échouer et apprendre

✅ On peut exprimer vulnérabilité

**Comment SM crée Psychological Safety :**

### **1. Modéliser la vulnérabilité**

❌ SM qui joue "l'expert qui sait tout"

"Je vais vous montrer la bonne façon"

✅ SM qui admet ne pas savoir

"C'est une situation que je n'ai jamais rencontrée.

Qu'en pensez-vous ? Explorons ensemble."

→ Donne permission à l'équipe d'admettre incertitude

### **2. Célébrer les erreurs (intelligentes)**

Developper : "J'ai mergé sur main par erreur, j'ai cassé prod"

❌ Réaction toxique :

"Comment as-tu pu faire ça ?! On a des process !"

✅ Réaction "healing" :

"Merci d'avoir signalé immédiatement. Comment peut-on

réparer rapidement ? Ensuite, qu'est-ce qu'on peut

apprendre pour éviter ça à l'avenir ?"

Post-incident :

"Cette erreur nous a permis de découvrir que notre

process de deploy manque de safeguards. On va l'améliorer."

→ Transforme erreur en amélioration

→ Pas de blame

→ Focus sur système, pas personne

### **3. Faciliter Rétrospectives "Healing"**

Après Sprint difficile (burnout, bugs, tensions) :

Format Retro "Healing" :

1. Reconnaissance : "Ce Sprint était dur. C'est ok de le dire."

2. Expression émotions :

"Utilisez ces 3 mots : Mad, Sad, Glad  
pour exprimer ce que vous ressentez"

3. Support mutuel :

"Qu'est-ce qu'on peut faire EN ÉQUIPE pour  
se sentir mieux au prochain Sprint ?"

4. Actions concrètes :

"1-2 actions MAX pour réduire stress"

5. Clôture positive :

"Une chose que tu apprécies chez un coéquipier"

→ Permet équipe de "guérir" collectivement

### **Situation Complexe PSM II :**

Scénario :

Votre équipe a vécu un incident de production majeur causant perte de données client. Le CEO a crié sur l'équipe en réunion publique. L'équipe est traumatisée, a peur de prendre des décisions, la créativité a disparu.

Un Developer senior a démissionné. Les autres sont en mode "CYA" (Cover Your Ass) - documentation excessive, pas de risques, paralysie décisionnelle.

Question : Comment en tant que SM, aidez-vous l'équipe à "guérir" ?

Réponse Modèle PSM II :

1. IMMÉDIAT : Parler au CEO (contexte séparé, pas devant équipe)

- Expliquer impact de son comportement
- Demander excuse publique (restaurer statut équipe)

2. AVEC L'ÉQUIPE : Session dédiée (pas retro normale)

- Reconnaître trauma : "Ce qui s'est passé n'était pas ok"
- Laisser exprimer : chacun partage ressenti (sans jugement)
- Normaliser : "C'est normal de se sentir ainsi après un tel événement"

3. RESTAURER CONFIANCE :

- Créer "safety net" : Plus de code reviews, pair programming
- Petits succès : Projets low-risk pour reconstruire confiance
- Célébrer courage : "Merci d'avoir essayé cette approche"

4. LONG-TERME :

- Blameless postmortem de l'incident (focus système, pas personnes)
- Améliorer processus (CI/CD, feature flags, rollback rapide)
- Coaching CEO sur leadership (si possible)

## 5. PRÉVENTION :

- Working agreements : Comment communiquer lors de crises
- Escalation path clair
- Célébrer failures intelligents (experimentation)

→ "Healing" est processus, pas événement ponctuel

---

### Caractéristique 📌 : AWARENESS (Conscience) 👁️

**Définition :** Conscience de soi et des autres. Comprendre les dynamiques, les patterns, l'environnement.

#### Pour le SM : 3 Types d'Awareness

##### 1. Self-Awareness (Conscience de Soi) 🧠

Le SM doit connaître :

- ✅ Ses forces et faiblesses
- ✅ Ses biais cognitifs
- ✅ Ses triggers émotionnels
- ✅ Son impact sur les autres
- ✅ Ses angles morts

Exemple :

"Je sais que je deviens directif quand je stresse.

Donc quand je sens pression, je fais une pause avant de parler en Daily Scrum."

#### Biais Cognitifs Courants chez SM :

Biais	Description	Danger pour SM	Mitigation
<b>Confirmation Bias</b>	Chercher info qui confirme nos croyances	"Je savais que cette équipe était paresseuse"	Chercher activement preuves contraires

Biais	Description	Danger pour SM	Mitigation
		(ignore preuves contraires)	
<b>Fundamental Attribution Error</b>	Blâmer personnes, ignorer contexte	"Il est incompetent" vs "Le système est cassé"	Toujours chercher facteurs systémiques
<b>Halo Effect</b>	Une qualité colore tout le reste	"Il est brillant en code donc son idée processus est forcément bonne"	Évaluer chaque situation indépendamment
<b>Anchoring</b>	Première info reçue ancre jugement	Première estimation devient "la vérité"	Chercher multiples perspectives
<b>Availability Bias</b>	Ce qui vient à l'esprit facilement semble plus probable	"Les bugs sont toujours causés par X" (car c'était le cas récemment)	Regarder données objectives

### Exercice Self-Awareness :

Après chaque événement Scrum, se demander :

1. Comment me suis-je senti ? (émotions)
2. Comment ai-je agi ? (comportements)
3. Quel a été mon impact ? (résultats)
4. Qu'est-ce que j'aurais pu faire différemment ?

Tenir journal de bord SM 📝

## 2. Team Awareness (Conscience d'Équipe) 👥

Le SM observe patterns :

- ✅ Dynamiques de pouvoir (qui influence décisions ?)
- ✅ Non-dits (tensions sous surface)
- ✅ Étapes de développement équipe (Tuckman)
- ✅ Santé émotionnelle collective

✅ Patterns de communication

## Modèle Tuckman - Stages de Développement d'Équipe

### 1. FORMING (Formation)

Comportements :

- Polis, formels
- Peu de confiance
- Dépendance au leader
- Incertitude sur rôles

Rôle SM :

- Établir normes et attentes
- Faciliter connaissance mutuelle
- Teaching (Scrum framework)
- Créer sécurité

Durée : 2-4 semaines

### 2. STORMING (Tempête)

Comportements :

- Conflits émergent
- Résistance aux normes
- Compétition pour influence
- Frustration, émotions fortes

Rôle SM :

- Faciliter conflits constructifs
- Ne PAS éviter tensions
- Clarifier rôles et boundaries

- Coaching intensif
- Rappeler objectif commun

Durée : 2-8 semaines (variable)

⚠️ Beaucoup d'équipes restent bloquées ici !

### 3. NORMING (Normalisation) 🧐

Comportements :

- Résolution de conflits efficace
- Consensus sur façons de travailler
- Respect mutuel établi
- Collaboration fluide

Rôle SM :

- Coaching léger
- Renforcer comportements positifs
- Faciliter amélioration continue
- Commencer à "lâcher prise"

Durée : 3-6 mois

### 4. PERFORMING (Performance) 🚀

Comportements :

- Auto-organisation naturelle
- High trust
- Résolution problèmes autonome
- Innovation et créativité
- Deliver valeur consistante

Rôle SM :

- Observer, minimale intervention
- Coaching stratégique
- Challenge l'équipe (stretch goals)
- Protéger des disruptions externes

Durée : Indéfinie (si équipe stable)

## 5. ADJOURNING (Séparation) 🙌

Quand équipe se dissout

Rôle SM :

- Célébrer accomplissements
- Faciliter retrospective finale
- Capturer learnings
- Support émotionnel (deuil)

### **Awareness en Action :**

Situation : Nouvelle Scrum Team, semaine 3

SM observe (Team Awareness) :

- Alice (senior) domine les discussions techniques
- Bob et Carol se taisent systématiquement
- Daily Scrum = Alice monologue, autres écoutent passivement
- Pas de désaccords exprimés (trop calme)
- Langage corporel : Bob croise bras quand Alice parle

Diagnostic : Équipe en début STORMING

- Hiérarchie informelle s'établit (Alice = alpha)
- Autres n'osent pas challenger
- Conflit latent (Bob frustré)

Action SM :

1. Changer format Daily : "Walk the board" au lieu de tour de table

→ Réduit dominance Alice

2. Rétrospective dédiée : "Working Agreements"

→ Établir règle : "Chacun parle égale durée"

3. 1-on-1 avec Bob :

SM : "J'ai remarqué que tu ne partages pas beaucoup  
tes idées. Qu'est-ce qui t'empêche ?"

Bob : "Alice a toujours raison, pourquoi parler ?"

SM : "Ton expertise est précieuse. Comment peut-on  
créer espace pour que tu la partages ?"

4. Coaching Alice :

"Tu es très compétente, mais j'ai observé que  
l'équipe participe moins. Que penses-tu qu'on  
pourrait faire pour encourager plus de voix ?"

→ Faire émerger sa propre prise de conscience

5. Team exercise : "Delegation Poker"

→ Clarifier qui décide quoi

→ Donner permission explicite de challenger

Résultat attendu :

- Conflit sain émerge (Storming continue)
  - Puis normalisation (Norming)
  - Équipe plus équilibrée
- 

### 3. Organizational Awareness (Conscience Organisationnelle) 🏢

Le SM comprend :

- ✓ Culture organisationnelle
- ✓ Politique et dynamiques de pouvoir
- ✓ Structures décisionnelles
- ✓ Impediments systémiques
- ✓ Résistance au changement (d'où vient-elle ?)

#### Exemple Organisational Awareness :

Scénario :

Vous êtes SM dans grande entreprise traditionnelle qui "adopte Scrum". Vous observez :

- Sprints existent, mais... :

- ✗ Sprint Planning = management assigne travail
- ✗ PO doit avoir approbation VP pour chaque story
- ✗ Releases nécessitent 5 approbations (CAB - Change Advisory Board)
- ✗ Équipe évaluée individuellement sur "output"
- ✗ Bonus liés à respect de deadlines fixées par management

- Culture observée :

- ✗ "Failure is not an option" (peur d'échouer)
- ✗ Micro-management généralisé

❌ Décisions top-down

❌ Silos départementaux forts

Organizational Awareness du SM :

🔍 Diagnostic :

"L'organisation a les MÉCANIQUES de Scrum mais pas les PRINCIPES. C'est du 'ScrumBut' / 'Zombie Scrum'."

Comprendre le WHY :

- Pourquoi management micro-manage ?

→ Peur de perdre contrôle (années de command & control)

- Pourquoi CAB existe ?

→ Incident majeur il y a 3 ans (trauma organisationnel)

- Pourquoi évaluations individuelles ?

→ Politique RH datant de 20 ans, jamais challengée

🎯 Stratégie (basée sur awareness) :

Court-terme (3-6 mois) :

1. Pas encore challenger CAB/évaluations (trop gros)

2. Focus micro-améliorations :

- Sprint Goal visible (pas juste liste tâches)

- Sprint Review avec stakeholders (montrer valeur)

- Metrics : valeur livrée vs output (éduquer management)

Moyen-terme (6-12 mois) :

3. Coalitions :

- Trouver sponsors dans middle management
- Communauté de pratique avec autres SMs
- Montrer résultats early adopters

4. Éduquer up :

- Présenter case studies externes
- Inviter consultants reconnus
- "Lunch & Learn" pour executives

Long-terme (1-2 ans) :

5. Systemic change :

- Proposer pilot program (1 équipe exemptée de CAB)
- Mesurer impact (time-to-market, qualité, morale)
- Scale progressivement si succès

⚠ Sans Organizational Awareness :

SM frustré qui dit "Ils ne font pas Scrum correctement !"

→ Burnout, démission

✅ Avec Organizational Awareness :

SM patient qui comprend contexte et joue long game

→ Transformation progressive, durable

---

**Caractéristique 5 : PERSUASION (Persuasion)** 💬

**Définition :** Convaincre par arguments et consensus, PAS par autorité ou coercition.

**Pour SM :**

SM n'a PAS d'autorité hiérarchique

Donc doit persuader :

- Équipe d'adopter pratiques
- PO de prioriser différemment
- Management de changer policies
- Organisation d'être plus agile

**Persuasion ≠ Manipulation** ⚠️

MANIPULATION (❌) :

- Cacher ses intentions
- Manipuler émotions
- Pour son propre bénéfice
- "Win-Lose"

PERSUASION (✅) :

- Intentions transparentes
- Arguments rationnels + émotionnels
- Pour bénéfice mutuel
- "Win-Win"

**Framework : Les 6 Principes de Persuasion (Robert Cialdini)** 📄

### **1. Reciprocity (Réciprocité)** 🎁

Principe : Les gens se sentent obligés de rendre une faveur

Application SM :

Exemple :

PO débordé, difficile d'avoir son temps

❌ Demande directe :

"J'ai besoin que tu sois plus disponible"

✅ Avec réciprocité :

1. D'abord, AIDER le PO :

"J'ai préparé un draft de stories basé sur ton dernier feedback. Ça peut te faire gagner du temps."

2. Puis, demander :

"En échange, pourrais-tu bloquer 30 min par jour pour questions de l'équipe ?"

→ PO plus enclin à accepter car il a reçu d'abord

## **2. Commitment & Consistency (Engagement et Cohérence)** 📄

Principe : Les gens veulent être cohérents avec leurs engagements publics

Application SM :

Exemple : Équipe ne respecte pas Definition of Done

❌ Imposer :

"À partir de maintenant, tout doit respecter DoD"

✅ Avec commitment :

1. Faciliter discussion équipe :

"Quelle qualité voulons-nous pour notre produit ?"

2. Co-créer DoD :

Équipe définit elle-même les critères

3. Rendre public :

DoD visible sur mur, signed par tous

4. Rappeler leur commitment :

"Vous aviez décidé que tests unitaires étaient  
essentiels. Que se passe-t-il ?"

→ Équipe respecte mieux car elle a choisi (ownership)

### **3. Social Proof (Preuve Sociale) 👥**

Principe : Les gens regardent ce que font les autres

Application SM :

Exemple : Management résiste à rétrospectives

("perte de temps")

❌ Argumenter seul :

"Les rétrospectives sont importantes"

✅ Avec social proof :

"J'ai discuté avec les 5 autres équipes qui font Scrum ici.

Toutes disent que la rétro est leur événement le plus  
valuable. Team A a réduit ses bugs de 40% grâce aux  
actions de rétro. Team B a amélioré sa vélocité de 25%.

De plus, Google, Spotify, et Amazon ont des rétrospectives

systématiques dans leurs équipes produit. Dans le dernier State of Agile Report, 89% des équipes high-performing font des rétros régulières."


→ Management convaincu par "tout le monde le fait"

#### **4. Authority (Autorité)**


Principe : Les gens respectent l'expertise et les credentials

Application SM :

Exemple : Convaincre executives d'investir dans automatisation

 Opinion personnelle :

"Je pense qu'on devrait automatiser les tests"

 Avec authority :

"Selon Martin Fowler (Chief Scientist ThoughtWorks),  
le ROI de l'automatisation de tests est de 400% sur 2 ans.

Une étude de Gartner montre que les organisations avec  
80%+ de tests automatisés ont 50% moins de bugs en production.

Jeff Bezos a dit : 'If you're going to do anything new or  
innovative, you have to be willing to be misunderstood.'  
Amazon investit massivement dans DevOps automation.

Ken Schwaber, co-créateur de Scrum, dit que l'automatisation  
fait partie de la Definition of Done moderne."

→ Arguments appuyés par autorités reconnues

## 5. Liking (Affinité) 😊

Principe : On est plus persuadé par les gens qu'on aime

Application SM :

Construire affinité par :

- ✅ Similarités : "Moi aussi j'ai travaillé 10 ans en waterfall"
- ✅ Compliments sincères : "Ton expertise technique impressionne l'équipe"
- ✅ Coopération : "Résolvons ce problème ensemble"
- ✅ Humour approprié
- ✅ Intérêt genuinement pour l'autre

Exemple :

Developper résiste pair programming

❌ Forcer :

"Le pair programming est obligatoire"

✅ Construire affinité d'abord :

SM apprend que Developper adore sci-fi

SM : "J'ai vu que tu lis Isaac Asimov. Moi aussi !

Foundation est génial."

[Discussion sur sci-fi, connexion humaine]

Puis plus tard, naturellement :

SM : "Tu sais, le pair programming c'est un peu comme avoir un co-pilote dans un vaisseau spatial. Asimov écrivait souvent sur l'importance des équipages collaboratifs..."

→ Plus ouvert à écouter car affinité établie

## **6. Scarcity (Rareté)** 🕒

Principe : Les gens veulent ce qui est rare ou limité

Application SM :

⚠️ À utiliser avec éthique !

Exemple : Convaincre équipe de faire Spike technique  
(research) maintenant

❌ "On devrait faire ce Spike"

✅ Avec scarcity :

"L'architecte principal qui connaît ce legacy system part à la retraite dans 2 mois. C'est notre dernière fenêtre pour faire ce Spike avec son expertise avant de perdre cette connaissance pour toujours.

Si on attend, on risque 6+ mois de galère à redécouvrir."

→ Urgence créée, équipe priorise le Spike

### **Combinaison des Principes :**

Situation PSM II :

Vous devez convaincre le VP Engineering d'arrêter de "parachuter" des features urgentes mid-Sprint sans consulter le PO.

Stratégie persuasive (multi-principes) :

#### **1. AUTHORITY :**

"Le Scrum Guide, créé par Ken Schwaber et Jeff Sutherland (pionniers Agile), stipule que seul le PO peut changer le scope du Sprint."

#### **2. SOCIAL PROOF :**

"Les 3 autres VPs (Sales, Product, Marketing) respectent ce processus. Votre équipe d'engineering est la seule où c'est différent."

#### **3. LIKING :**

"J'apprécie votre passion pour livrer de la valeur rapidement. C'est exactement pourquoi Scrum existe."

#### **4. RECIPROCITY :**

"Je vais créer un processus 'fast-track' où les urgences réelles peuvent être évaluées en < 24h avec le PO. En échange, respectons le Sprint pour les non-urgences."

## 5. COMMITMENT :

"Pouvez-vous vous engager à essayer ce processus pendant 2 Sprints et ensuite on évalue ensemble ?"

## 6. SCARCITY :

"Les 2 meilleurs Developers envisagent de partir à cause du chaos constant. On a une fenêtre courte pour améliorer avant de les perdre."

→ Approche multi-angles, difficile de résister

---

## Caractéristique : CONCEPTUALIZATION (Conceptualisation)

**Définition :** Capacité à voir au-delà des opérations quotidiennes, penser de façon systémique et visionnaire.

### Pour SM :

Ne pas juste "firefighter" (éteindre feux)

Mais voir patterns, anticiper, penser long-terme

### Tactical vs Strategic Thinking

TACTICAL (Court-terme, réactif) :

"Cette story est bloquée, je vais la débloquer"

"Ce conflit nécessite médiation"

"Ce Daily prend trop de temps, je vais timeboxer"

STRATEGIC (Long-terme, proactif) :

"Pourquoi avons-nous tant de blocages ? Problème systémique ?"

"Ce conflit révèle un problème structurel dans l'équipe"

"Pourquoi nos Dailies sont inefficaces ? Problème de format ou symptôme d'un manque d'alignement plus profond ?"


## Systems Thinking (Pensée Systémique)

**Principe :** Comprendre que tout est interconnecté. Un problème "ici" a souvent des causes "là-bas".

### Exemple Classique :

SYMPTÔME observé :

"Les Developers livrent des stories avec plein de bugs"

Réaction NON-systémique (  ) :

"Les Developers sont négligents, ils doivent faire plus attention"

→ Blâme les individus

Réaction SYSTÉMIQUE (  ) :

SM analyse le SYSTÈME :

1. Pourquoi bugs ?

→ Pas de tests unitaires

2. Pourquoi pas de tests ?

→ "Pas le temps"

3. Pourquoi pas le temps ?

→ Pression pour "finir tout le Sprint Backlog"

4. Pourquoi cette pression ?

→ PO overcommit chaque Sprint

5. Pourquoi PO overcommit ?

→ Management demande "plus de features"

6. Pourquoi management demande plus ?

→ Métriques = # features livrées (pas qualité)

7. Pourquoi ces métriques ?

→ Culture organisationnelle "output over outcome"

ROOT CAUSE : Système de métriques organisationnel

SOLUTION SYSTÉMIQUE :

- Court-terme : DoD stricte (tests obligatoires)
- Moyen-terme : Éduquer PO sur vélocité durable
- Long-terme : Changer métriques organisationnelles  
(business value au lieu de # features)

→ Traite la CAUSE, pas juste le SYMPTÔME

**Modèle : Iceberg Model** 

[Événement visible]

===== Surface

[Patterns/Tendances]

|

[Structures Sous-jacentes]

|

[Mental Models/Culture]

NIVEAU 1 - ÉVÉNEMENTS (Ce qu'on voit) :

"Un Developer a quitté l'équipe"

## NIVEAU 2 - PATTERNS (Répétitions) :

"3 Developers ont quitté en 6 mois"

## NIVEAU 3 - STRUCTURES (Systèmes) :

"Process de recrutement attrape mauvais profils"

"Pas d'onboarding structuré"

"Compensation non-compétitive"

## NIVEAU 4 - MENTAL MODELS (Croyances) :

"Developers sont remplaçables"

"Output > Well-being"

"Court-terme > Long-terme"

Intervention SM par niveau :

Niveau 1 : Exit interview avec le Developer

Niveau 2 : Analyse de rétention, patterns de départ

Niveau 3 : Améliorer onboarding, comp, conditions

Niveau 4 : Changer culture, éduquer leadership sur  
valeur de rétention des talents

→ SM conceptualise à tous les niveaux

## **Anticipation et Vision 🧠**

SM Conceptualisateur pose :


"Si on continue comme ça, où sera-t-on dans 6 mois ?"

"Quel pattern émerge ?"

"Quelle est la prochaine étape logique de maturité ?"

### Exemple :

Observation SM (mois 1-3) :

- Équipe livre features régulièrement 
- Mais vélocité plafonne à 25 SP depuis 3 Sprints
- Temps passé en bug fixing augmente (10% → 20% → 30%)
- Definition of Done respectée... mais minimale

Conceptualisation :

"On est en train de créer une dette technique qui va nous rattraper. Dans 6 mois, on passera 50%+ du temps en bug fixing et vélocité chutera à 15 SP. Le code deviendra unmaintainable."

Vision :

"Il faut investir MAINTENANT dans qualité technique (tests auto, refactoring, code quality) pour éviter ce futur."

Action Proactive (pas réactive) :

1. Faciliter discussion avec PO sur trade-offs
2. Proposer "20% time" pour tech health chaque Sprint
3. Améliorer DoD (coverage, code review, etc.)
4. Former équipe sur clean code practices

Résultat :

Dette technique évitée AVANT la crise, pas pendant

**Définition :** Anticiper conséquences futures des décisions présentes. Apprendre du passé pour prédire futur.

**Pour SM :**

"Si on fait X maintenant, qu'est-ce qui va se passer ?"

"Cette décision va créer quel problème dans 3 Sprints ?"

"J'ai vu ce pattern avant, voici ce qui va arriver..."

**Différence Conceptualization vs Foresight :**

CONCEPTUALIZATION :

Voir patterns systémiques, penser abstraitement

FORESIGHT :

Prédire conséquences spécifiques futures

**Application Pratique :**

Situation :

PO veut "juste cette fois" ajouter 3 features urgentes

mid-Sprint sans retirer d'autres items.

SM SANS Foresight (  ) :

"Ok, juste cette fois"

SM AVEC Foresight (  ) :

"J'ai vu ce pattern dans mes équipes précédentes.

Voici ce qui va se passer :

Sprint actuel :

- Équipe stressée, tente de tout finir
- Quality souffre (pas temps pour tests)
- Probablement rien n'est vraiment 'Done'

Sprint suivant :

- Équipe fatiguée, motivation basse
- Bugs du Sprint précédent remontent
- Vitesse chute de 30-40%
- Stakeholders mécontents (features buggées)

Dans 3 Sprints :

- Pattern établi : PO ajoute toujours mid-Sprint
- Équipe ne prend plus Sprint Planning au sérieux
- Prédicibilité disparaît
- 'Scrum' devient chaos

Dans 6 mois :

- Turnover : Developers partent (burnout)
- Qualité produit terrible
- Organisation abandonne Scrum

Alternative :

Si c'est VRAIMENT urgent → annuler Sprint, re-plan

Sinon → attendre prochain Sprint Planning

Qu'en penses-tu ?"

→ SM peint le futur pour aider décision éclairée

**Technique : Pre-Mortem** 🧠

Concept (opposé de post-mortem) :

Imaginer que le projet a ÉCHOUÉ dans le futur,

puis travailler backwards pour identifier causes

Application Sprint Planning :

SM : "Ok, on a notre Sprint Goal et Sprint Backlog.

Faisons un 'Pre-Mortem'.

Imaginons qu'on est à la Sprint Review dans 2 semaines.

Le Sprint Goal n'a PAS été atteint. Échec total.

Qu'est-ce qui s'est passé ?"

Équipe brainstorm :

- "La dépendance avec l'équipe Backend n'a pas été résolue"
- "On a sous-estimé la complexité du login OAuth"
- "Marie est en congé la semaine prochaine, on l'avait oublié"
- "L'environnement de staging va être down mardi-mercredi"

Action :

→ Mitiger ces risques AVANT qu'ils deviennent réels

→ Ajuster Sprint Backlog en conséquence

→ Foresight transformé en action préventive

---

**Caractéristique 8 : STEWARDSHIP (Intendance) 🌍**

**Définition :** Engagement envers le bien-être de l'équipe, l'organisation et la société.  
Penser au-delà de l'intérêt personnel.

**Pour SM :**

Pas "qu'est-ce qui est bon pour MOI ?"

Mais "qu'est-ce qui est bon pour l'ÉQUIPE, le PRODUIT,  
l'ORGANISATION à long-terme ?"

### **Manifestations Stewardship :**

#### **1. Développer les Autres (pas se rendre indispensable) 🌱**

❌ Mauvais SM (Crée dépendance) :

"Je vais résoudre tous vos problèmes"

"Venez me voir pour toute décision"

"Je suis essentiel, sans moi ça ne marche pas"

→ Équipe dépendante, SM devient bottleneck

✅ Bon SM (Développe autonomie) :

"Comment pourriez-vous résoudre ce problème ?"

"Qu'avez-vous essayé ? Quelle est votre hypothèse ?"

"Vous n'avez pas besoin de moi pour décider ça"

→ Équipe autonome, SM devient optionnel (succès !)

### **Paradoxe du SM Excellent :**

Le meilleur SM rend son rôle obsolète dans l'équipe

Signes de succès :

✅ Équipe auto-facilite ses événements

✅ Équipe résout ses impediments

✅ Équipe améliore continuellement sans prompting

✅ SM peut se concentrer sur coaching org/PO

"Mon job est de me mettre au chômage dans cette équipe"

→ Puis passer à une autre équipe qui a besoin de help

## 2. Protéger Long-Terme vs Court-Terme ⏳

Situation fréquente :

Management : "Skip les tests cette fois, on doit livrer"

❌ SM qui pense court-terme :

"Ok, juste cette fois"

→ Dette technique, problèmes futurs

✅ SM Steward (pense long-terme) :

"Je comprends l'urgence. Mais skipping tests crée

dette qui ralentira toutes livraisons futures.

Dans 3 mois, on ne pourra plus livrer rapidement.

Alternative : Réduisons scope mais gardons qualité.

Qu'est-ce qui est le MINIMUM viable pour cette release ?"

→ Protège santé long-terme du produit

## 3. Éthique et Intégrité ⚖️

SM fait face à dilemmes éthiques :

Exemple 1 : Pression pour manipuler métriques

Management : "Augmente la vélocité sur papier pour le board meeting"

SM Steward :

"Non. La vélocité est un outil de planification, pas une

métrique de performance. La manipuler détruit sa valeur

et trompe l'organisation. Je ne le ferai pas."

→ Refuse, même si risque son poste

Exemple 2 : Demande de cacher problèmes

Manager : "Ne mentionne pas ces bugs en Steering Committee"

SM Steward :

"Transparence est une valeur Scrum fondamentale. Cacher les problèmes empêche de les résoudre. Je dois partager la réalité."

→ Transparence > Politique

#### **4. Sustainability (Durabilité) 🌱**

Veiller au bien-être humain, pas juste output

Signes de SM non-Steward :

❌ "On va faire des heures sup pour finir le Sprint"

❌ "Travaillez le weekend pour la deadline"

❌ "Performance = heures travaillées"

SM Steward :

✅ Surveille signes de burnout

✅ Encourage work-life balance

✅ "40h/semaine maximum, qualité pas quantité"

✅ Prône rythme soutenable (Sustainable Pace - Agile)

Citation Agile Manifesto :

"Agile processes promote sustainable development.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely."

→ Marathon, pas sprint (ironique pour "Sprints" !)

## 5. Legacy (Héritage) 📄

SM Steward pense :

"Quel héritage je laisse ?"

"L'équipe sera-t-elle meilleure après mon départ ?"

"Ai-je développé futurs servant leaders ?"

Actions concrètes :

- ✅ Documenter learnings (playbooks, retros)
- ✅ Mentorer futurs SMs
- ✅ Créer communautés de pratique
- ✅ Partager échecs et succès ouvertement
- ✅ Former d'autres à faciliter/coacher

Objectif :

Pas "J'étais un grand SM"

Mais "J'ai créé une culture où Scrum prospère sans moi"

---

**Caractéristique ? : COMMITMENT TO GROWTH (Engagement Envers la Croissance) 📈**

**Définition :** Engagement profond à la croissance personnelle et professionnelle des membres de l'équipe.

**Pour SM :**

Chaque personne a du potentiel

Rôle SM = libérer ce potentiel

Pas juste "faire le job", mais GRANDIR

### **Growth Mindset (Carol Dweck)** 🧠

FIXED MINDSET :

"L'intelligence est fixe"

"Je suis bon ou mauvais en X"

"L'échec prouve mon incompetence"

GROWTH MINDSET :

"L'intelligence se développe"

"Je peux apprendre X avec effort"

"L'échec est opportunité d'apprendre"

### **SM Cultive Growth Mindset :**

Developper : "Je suis nul en frontend"

❌ Fixed mindset response :

"Ok, alors fais que du backend"

→ Renforce limitation

✅ Growth mindset response :

"Tu n'es pas encore fort en frontend."

Qu'est-ce qui te bloquerait d'apprendre ?

Veux-tu paier avec Sarah qui excelle en React ?"

→ Encourage croissance

### **Créer Opportunités de Croissance** 🚀

#### **1. Stretch Assignments**

Donner tâches juste au-delà de confort zone

Exemple :

Junior Developer a toujours fait des bugs fixes

SM : "On a une feature complexe ce Sprint.

Veux-tu la prendre en pair avec un senior ?

Tu vas apprendre énormément."

→ Challenge avec support (zone apprentissage optimal)

## **2. Rotation de Rôles**

Permettre essayer différentes responsabilités

Exemples :

- Developer facilite Sprint Planning (avec coaching SM)
- QA écrit code (pair programming)
- Backend essaie Frontend
- Chacun facilite une Retrospective à tour de rôle

→ Développe T-shaped skills

→ Empathie inter-rôles

→ Réduit dependencies

## **3. Learning Time Explicite**

SM champion :

"20% du temps Sprint = apprentissage"

Formes :

- Lunch & Learns
- Coding dojos

- Book clubs
- Conférences (temps et budget)
- Certifications
- Expérimentations techniques

→ Apprentissage n'est pas "si on a le temps"  
mais priorité explicite

#### **4. Safe-to-Fail Experiments**

Créer environnement où échec = apprentissage

Exemple :

Équipe veut essayer Mob Programming

SM facilite :

1. "Expérimentation" : Essai 2 Sprints
2. Mesures claires : Qualité, vitesse, satisfaction équipe
3. Retrospective dédiée : Learnings
4. Décision collective : Continuer ou non

Si échec :

✖ "Vous avez échoué"

✔ "Qu'avez-vous appris ?"

→ Innovation encouragée

**Feedback pour la Croissance** 

**Feedback Model : SBI (Situation-Behavior-Impact)** 

Structure :

1. SITUATION : Contexte spécifique

2. BEHAVIOR : Comportement observé

3. IMPACT : Impact de ce comportement

Exemple feedback positif :

"Situation : Pendant le Sprint Planning hier,

Behavior : tu as posé cette question 'Pourquoi on fait  
cette feature ?' au PO,

Impact : ça nous a fait réaliser qu'on avait mal compris  
le besoin utilisateur. On a économisé 2 semaines  
de travail inutile. Merci !"

→ Renforce comportement, développe confiance à questionner

Exemple feedback constructif :

"Situation : Pendant le Daily ce matin,

Behavior : tu as interrompu Alice 3 fois alors qu'elle  
expliquait son impediment,

Impact : elle n'a pas pu finir, et l'équipe n'a pas compris  
le blocage. Ça nous empêche de l'aider.  
Comment pourrais-tu faire différemment demain ?"

→ Spécifique, actionable, pas attaque personnelle

### **Coaching de Carrière** 🎓

SM a conversations régulières (hors Scrum events) :

Questions :

"Où veux-tu être dans 2 ans ?"

"Qu'est-ce qui t'excite professionnellement ?"

"Quelles compétences veux-tu développer ?"

"Comment puis-je t'aider à y arriver ?"

Actions SM :

- ✓ Connecter avec mentors
- ✓ Recommander formations
- ✓ Créer opportunités alignées avec aspirations
- ✓ Supporter participation conférences/meetups

→ SM investit dans le futur de chaque personne

---

### **Caractéristique 10 : BUILDING COMMUNITY (Construire la Communauté) 🏠**

**Définition :** Créer un sentiment d'appartenance, connecter les gens, bâtir des relations au-delà du travail.

**Pour SM :**

Équipe ≠ juste collègues qui travaillent ensemble

Équipe = Communauté qui se soucie les uns des autres

**Importance Communauté 🤝**

**Recherche (Project Aristotle - Google) :**

Google a étudié 180 équipes pour comprendre ce qui rend une équipe high-performing.

Résultat #1 (de loin) : Psychological Safety

Mais aussi : Sens d'appartenance, relations interpersonnelles

Équipes avec forte communauté :

- ✓ 50% plus productives
- ✓ 75% moins de turnover

- ✅ Innovation 2x supérieure
- ✅ Satisfaction 3x plus élevée

## **Comment SM Bâtit Communauté** 🏠

### **1. Rituels Sociaux**

Au-delà des Scrum events "business" :

Exemples :

- Coffee roulette : Café random avec coéquipier chaque semaine
- Team lunch mensuel (sur budget équipe)
- Célébrations : Anniversaires, succès, milestones
- Fun retrospectives : Escape room, bowling
- Show & Tell : Passions personnelles (pas juste travail)

→ Relations humaines, pas juste professionnelles

### **2. Connaître les Personnes** 👤

SM investit temps pour VRAIMENT connaître chacun :

Au-delà de "Salut, ça va ?" :

- ✅ Leurs passions (hobbies, sports, arts)
- ✅ Leurs familles (enfants, animaux)
- ✅ Leurs rêves professionnels
- ✅ Leurs challenges personnels (si partagent)
- ✅ Leurs valeurs, ce qui compte pour eux

Utilité :

1. Connexion humaine authentique
2. Comprendre motivations profondes

3. Adapter coaching à chaque personne

4. Créer espace de vulnérabilité saine

Exemple :

SM sait que Developer A est passionné de montagne

Sprint Review difficile, équipe démoralisée

SM à Developer A :

"Tu m'as dit que tu fais de l'alpinisme. Parfois la montée est dure, mais la vue au sommet vaut le coup. On a eu un Sprint difficile, mais regarde le chemin parcouru depuis 3 mois. Continue l'ascension."

→ Métaphore personnelle, impactante

### 3. Célébrer Ensemble 🎉

Succès ET échecs intelligents

Succès :

- ✅ Release majeure → Team dinner
- ✅ Atteinte Product Goal → Champagne 🍾
- ✅ 6 mois sans incident prod → Trophy 🏆
- ✅ Amélioration vélocité 50% → Bonus équipe

"Échecs" intelligents :

- ✅ Experiment raté mais learnings → "Learning pizza"
- ✅ Bug trouvé avant prod → "Bug bounty"

✅ Quelqu'un admet grosse erreur → "Courage award"

→ Renforce culture apprentissage et cohésion

#### 4. Traditions d'Équipe 🗒️

Créer identité unique à l'équipe

Exemples :

- Nom d'équipe choisi collectivement
- Logo / mascotte
- Inside jokes
- "Rubber duck" pour code review
- Playlist Spotify collaborative
- Mur de photos équipe
- "Quote board" avec phrases mémorables

→ "NOTRE équipe" (pas juste "une équipe")

#### 5. Conflicts = Opportunities 📌

Conflit ≠ mauvais pour communauté

Conflit évité = communauté superficielle

SM facilite "healthy conflict" :

Patrick Lencioni - 5 Dysfunctions of Teams :

1. Absence of Trust
2. Fear of Conflict ← SM aide surmonter
3. Lack of Commitment
4. Avoidance of Accountability
5. Inattention to Results

SM crée sécurité pour désaccords :

"On peut ne pas être d'accord ET se respecter"

"Les meilleures idées émergent du débat constructif"

"Attaquer les idées, pas les personnes"

Working Agreement :

"On s'engage à exprimer nos désaccords directement,  
avec respect, pour trouver meilleures solutions ensemble"

→ Communauté FORTE permet conflict sain

## **6. Communauté au-delà de l'Équipe**

SM connecte équipe avec :

- Autres Scrum Teams (cross-pollination)
- Communauté de pratique Scrum org
- Meetups / Conférences externes
- Open source communities
- User groups

Bénéfices :

- ✓ Idées fraîches
- ✓ Benchmarking
- ✓ Network
- ✓ Sentiment d'appartenance plus large

SM facilite :

- "Lunch & Learn" inter-équipes
- Scrum Master community of practice

- Sponsoring participation conférences
  - Contributions open source sur projet side
- 

## 2.2 Récapitulatif : Les 10 Caractéristiques

### Checklist Self-Assessment

Rate yourself 1-5 sur chaque caractéristique : 1 = Débutant, 5 = Master

- [ ] 1. LISTENING : J'écoute au niveau 3 (global)
- [ ] 2. EMPATHY : Je comprends vraiment les perspectives
- [ ] 3. HEALING : Je crée safety psychologique
- [ ] 4. AWARENESS : Self/Team/Org awareness forte
- [ ] 5. PERSUASION : J'influence sans autorité
- [ ] 6. CONCEPTUALIZATION : Je pense systémique
- [ ] 7. FORESIGHT : J'anticipe conséquences futures
- [ ] 8. STEWARDSHIP : Je protège long-terme
- [ ] 9. COMMITMENT TO GROWTH : Je développe les autres
- [ ] 10. BUILDING COMMUNITY : Je crée appartenance

### Développement Continu

Pour chaque caractéristique < 4 :

1. Identifier 1 action concrète ce mois
2. Pratiquer délibérément
3. Demander feedback
4. Ajuster
5. Répéter

Servant Leadership = pratique de toute une vie

---

## 2.3 Servant Leadership vs Traditional Leadership

### Comparaison Détaillée

Aspect	Traditional Leader	Servant Leader (SM)
Autorité	Top-down	Bottom-up
Focus	Résultats	Personnes + Résultats
Style	Diriger	Servir
Décisions	Décide seul	Facilite consensus
Communication	"Voici ce qu'on va faire"	"Qu'en pensez-vous ?"
Problèmes	Résout	Aide équipe à résoudre
Succès	"J'ai réussi"	"Ils ont réussi"
Échec	"Ils ont échoué"	"Comment puis-je aider ?"
Développement	Facultatif	Priorité centrale
Relation	Transactionnelle	Transformationnelle

---

### Question Exam PSM II :

Scénario :

Votre équipe livre régulièrement mais la qualité baisse.

Bugs en production augmentent. Vous savez exactement quelle est la solution technique (automatiser tests, refactoring modules clés, adopter TDD).


Deux approches possibles :

A. Dire à l'équipe : "Voici le plan technique à suivre pour résoudre les problèmes qualité. Je vais créer les tickets, vous les implémentez."

B. Faciliter une session où l'équipe analyse les causes des bugs, explore des solutions, et décide collectivement

des actions à prendre. Même si ça prend plus de temps  
et qu'ils arrivent potentiellement à une solution  
sous-optimale.

Quelle approche est la PLUS alignée avec Servant Leadership ?

Réponse : B 

Pourquoi :

- Servant Leader développe autonomie équipe
- Solutions co-crées = meilleur ownership
- Équipe apprend (croissance), pas juste exécute
- Même si sous-optimale court-terme, meilleur long-terme
- SM résiste à tentation de "sauver" l'équipe

Piège A :

- Plus rapide, plus "efficient"
- Mais crée dépendance
- Équipe ne développe pas capacité résolution problèmes
- SM devient bottleneck décisionnel

---

 **Prêt pour la Suite ? On attaque le Coaching Professionnel en détail ?**

(Suite : Module 3 - Coaching Professionnel : 100+ pages avec frameworks, techniques, exemples, exercices...)