Parcours C — MCP Server & Outils Déterministes

Adservio | Dr Olivier Vitrac

2026-02-03

Contents

1 Parcours C — MCP Server & Outils Déterministes **1**
1.1 Contexte & Enjeu . . . 2
1.2 Objectif Pédagogique . . . 2
1.3 Cahier des Charges Minimal . . . 2
1.3.1 Outils à Implémenter . . . 2
1.3.2 Caractéristiques des Outils . . . 2
1.3.3 Scénario Agent . . . 3
1.3.4 Scope Minimal Viable . . . 3
1.4 Contraintes Non-Négociables . . . 3
1.4.1 Ce qui est Interdit . . . 3
1.5 Auditabilité & Traçabilité . . . 3
1.5.1 Format de Trace Attendu . . . 3
1.5.2 Éléments à Tracer . . . 4
1.6 Livrables Attendus . . . 4
1.6.1 Structure Suggérée . . . 5
1.7 Critères d'Évaluation . . . 5
1.7.1 Barème . . . 5
1.8 Pièges & Anti-Patterns . . . 5
1.8.1 Piège 1 : L'outil qui ment . . . 5
1.8.2 Piège 2 : Le schéma vague . . . 5
1.8.3 Piège 3 : L'agent qui boucle . . . 6
1.8.4 Piège 4 : L'outil omnipotent . . . 6
1.8.5 Piège 5 : La trace cosmétique . . . 6
1.9 Retour Critique & Limites de l'Agent . . . 6
1.10 Questions Croisées (Préparation Demo Day) . . . 6
1.10.1 Vers le parcours A (Testing) . . . 6
1.10.2 Vers le parcours B (RAG) . . . 6
1.10.3 Intégration Future . . . 6
1.11 Ressources . . . 6

1 Parcours C — MCP Server & Outils Déterministes

"La vraie agentivité commence ici"

1.1 Contexte & Enjeu

Un LLM qui “fait des choses” sans outils externes est un LLM qui *simule* des actions. Il génère du texte qui *ressemble* à une exécution, mais rien ne se passe réellement.

La vraie agentivité requiert une séparation claire : - **Raisonnement** : le LLM décide *quoi* faire - **Exécution** : des outils déterministes *font* les choses - **Autorité** : un humain valide les décisions critiques

MCP (Model Context Protocol) formalise cette séparation. Un serveur MCP expose des outils avec des schémas stricts. L’agent appelle, l’outil exécute, l’agent interprète.

Enjeu pour Adservio : Comprendre que “agent” ≠ “LLM avec pouvoirs”, mais “LLM + outils + contrôle”.

1.2 Objectif Pédagogique

À l’issue de ce parcours, vous serez capables de :

1. **Concevoir** des outils déterministes avec schémas stricts
2. **Implémenter** un serveur MCP fonctionnel
3. **Connecter** un agent (Claude) à vos outils
4. **Tracer** les appels et décisions de l’agent
5. **Distinguer** ce que l’agent *décide* de ce qu’il *exécute*

1.3 Cahier des Charges Minimal

1.3.1 Outils à Implémenter

Créez un serveur MCP exposant **au minimum 4 outils** :

Outil	Description	Entrée	Sortie
`run_tests`	Lance pytest sur un chemin	`path: str`	`{passed, failed, errors}`
`lint_code`	Analyse statique (flake8/ruff)	`path: str`	`{issues: [...]}`
`parse_logs`	Extrait les erreurs d’un log	`log_path: str, pattern: str`	`{matches: [...]}`
`compute_metric`	Calcule une métrique code	`path: str, metric: str`	`{value: float}`

1.3.2 Caractéristiques des Outils

Chaque outil doit être :

1. **Déterministe** : mêmes entrées → mêmes sorties
2. **Documenté** : schéma JSON complet (types, descriptions)
3. **Borné** : timeout, limites de ressources
4. **Tracé** : chaque appel loggé avec entrées/sorties

1.3.3 Scénario Agent

Implémentez **au moins un scénario** où l'agent :

1. Reçoit une tâche (ex: "Analyse la qualité du code dans /src")
2. Décide quels outils appeler et dans quel ordre
3. Appelle les outils via MCP
4. Interprète les résultats
5. Produit une synthèse

1.3.4 Scope Minimal Viable

- 4 outils fonctionnels
- 1 scénario complet exécuté
- Trace de tous les appels

1.4 Contraintes Non-Négociables

Contrainte	Justification
Outils déterministes	Pas de random, pas de side-effects cachés
Schémas stricts	JSON Schema complet pour chaque outil
Pas de simulation	L'outil *exécute*, il ne *prétend* pas
Trace systématique	Chaque appel d'outil loggé
Un livrable sans trace exploitable est considéré comme non livré	Règle d'or Adservio

1.4.1 Ce qui est Interdit

- Laisser l'agent "simuler" l'exécution d'un outil
- Outils sans schéma ou avec schéma incomplet
- Exécution sans timeout (risque de hang)
- Outils avec effets de bord non documentés

1.5 Auditabilité & Traçabilité

> Toute action de l'agent doit produire une trace exploitable par un humain : décisions, appels d'outils, hypothèses, erreurs, sorties intermédiaires.

1.5.1 Format de Trace Attendu

```
{
  "session_id": "uuid",
  "task": "Analyse la qualité du code dans /src",
  "steps": [
    {
      "step_id": 1,
      "type": "reasoning",
      "content": "Je vais d'abord lancer le linter pour identifier les problèmes évidents",
      "timestamp": "2026-02-02T14:30:00Z"
```

```
    },
    {
      "step_id": 2,
      "type": "tool_call",
      "tool": "lint_code",
      "input": {"path": "/src"},
      "output": {"issues": [{"file": "main.py", "line": 42, "msg": "unused import"}]},
      "duration_ms": 1234,
      "timestamp": "2026-02-02T14:30:01Z"
    },
    {
      "step_id": 3,
      "type": "reasoning",
      "content": "1 issue trouvé. Je vais maintenant calculer la complexité cyclomatique.",
      "timestamp": "2026-02-02T14:30:02Z"
    },
    {
      "step_id": 4,
      "type": "tool_call",
      "tool": "compute_metric",
      "input": {"path": "/src", "metric": "cyclomatic_complexity"},
      "output": {"value": 12.5},
      "duration_ms": 567,
      "timestamp": "2026-02-02T14:30:03Z"
    }
  ],
  "conclusion": "Code globalement correct, 1 import inutilisé, complexité moyenne acceptable.
}
```


1.5.2 Éléments à Tracer

Élément	Obligatoire	Description
`tool`	Oui	Nom de l'outil appelé
`input`	Oui	Paramètres exacts
`output`	Oui	Résultat retourné
`duration_ms`	Oui	Temps d'exécution
`timestamp`	Oui	Horodatage précis
`reasoning`	Recommandé	Pourquoi cet outil, pourquoi maintenant
`error`	Si applicable	Message d'erreur complet

1.6 Livrables Attendus

Livrable	Format	Description
Serveur MCP	Python	Code du serveur avec outils
Schémas d'outils	JSON	Définitions complètes des outils
Script scénario	Python	Orchestration agent + outils
Traces	JSON	Logs de tous les appels
Rapport	Markdown	Analyse des décisions de l'agent

1.6.1 Structure Suggérée

```
mcp_server/
├── server.py             # Serveur MCP principal
├── tools/
│   ├── __init__.py
│   ├── run_tests.py
│   ├── lint_code.py
│   ├── parse_logs.py
│   └── compute_metric.py
├── schemas/
│   └── tools.json        # Schémas JSON des outils
├── scenarios/
│   └── quality_check.py  # Scénario exemple
├── traces/
│   └── session_*.json    # Traces d'exécution
└── report.md             # Rapport final
```

1.7 Critères d'Évaluation

Critère	Poids	Indicateurs
Outils fonctionnels	25%	4 outils qui s'exécutent correctement
Schémas complets	15%	Types, descriptions, exemples
Déterminisme	20%	Mêmes entrées → mêmes sorties, vérifié
Scénario agent	20%	Agent utilise les outils de manière cohérente
Traçabilité	20%	Trace complète et exploitable

1.7.1 Barème

- **Insuffisant** : Outils partiels ou non déterministes
- **Passable** : Outils fonctionnels mais scénario incomplet
- **Satisfaisant** : Scénario complet avec trace
- **Excellent** : Outils robustes, trace exemplaire, insights sur les décisions agent

---

1.8 Pièges & Anti-Patterns

1.8.1 Piège 1 : L'outil qui ment

L'outil retourne un résultat mais ne fait pas vraiment l'action. → **Solution** : Vérifier l'effet réel (fichier créé, process lancé, etc.).

1.8.2 Piège 2 : Le schéma vague

```json
{"input": "any", "output": "any"}
```

→ **Solution** : Types précis, contraintes explicites, exemples.

1.8.3 Piège 3 : L'agent qui boucle

L'agent appelle le même outil 50 fois sans progresser. → **Solution** : Limiter le nombre d'appels, détecter les boucles.

1.8.4 Piège 4 : L'outil omnipotent

Un seul outil "do_everything" qui cache la complexité. → **Solution** : Outils atomiques, responsabilité unique.

1.8.5 Piège 5 : La trace cosmétique

Trace présente mais inexploitable (pas de timestamps, pas de contexte). → **Solution** : Suivre le format défini, vérifier la rejouabilité.

1.9 Retour Critique & Limites de l'Agent

À compléter **obligatoirement** à la fin du parcours :

1. **Qu'est-ce que l'agent a *semblé* bien décider mais n'a pas réellement optimisé ?**
2. **Où lui avez-vous fait confiance à tort sur le choix d'outils ?**
3. **Quelle hypothèse *vous* avez faite sur l'orchestration qui s'est révélée fausse ?**
4. **Qu'interdiriez-vous à cet agent en production ?**
5. **Qu'est-ce qui vous a le plus surpris dans ses décisions ?**

1.10 Questions Croisées (Préparation Demo Day)

1.10.1 Vers le parcours A (Testing)

- Comment votre outil `run_tests` pourrait-il être utilisé par un agent de testing ?
- Quels autres outils ajouteriez-vous pour supporter un agent de test complet ?

1.10.2 Vers le parcours B (RAG)

- Comment exposer un index RAG comme outil MCP ?
- Un outil `search_docs` serait-il déterministe ?

1.10.3 Intégration Future

- Comment versionner vos schémas d'outils ?
- Comment gérer les mises à jour d'outils sans casser les agents existants ?

1.11 Ressources

- MCP Specification
- MCP Python SDK
- Claude Desktop MCP

- JSON Schema

Parcours C — MCP Server & Outils Déterministes — Adservio Workshop