

logiCells

Guide de modernisation des applications et de la connaissance métier

Retrouver, structurer et exploiter
la connaissance métier

Du comportement historique à une
connaissance explicite et vérifiée

Note éditoriale

Moderniser une application ne consiste pas simplement à remplacer un logiciel. Il s'agit de retrouver les décisions, les distinctions et les obligations métier qu'une organisation a progressivement intégrées dans ses technologies et ses pratiques opérationnelles. Ce guide propose une méthode rigoureuse pour expliciter cette connaissance et en faire un actif gouverné de façon indépendante.

Le cas fil rouge de gestion des commandes est fictif. Tous les exemples sont indépendants des technologies : ils illustrent une possible spécification cible et ne constituent ni des affirmations sur un produit exécutable, ni la promesse d'une capacité d'extraction automatique. Leur objectif est de rendre la méthode suffisamment concrète pour être discutée, appliquée et testée.

Deux questions structurent l'ouvrage : *qu'est-ce qu'une connaissance métier structurée ? Et comment rendre visible, justifiable et utile, dans un programme de modernisation, la connaissance métier enfouie dans une application existante ?*

Portée du guide. Méthodologie et exemples illustratifs : ce document ne constitue pas un manuel de mise en œuvre du produit.

ÉDITION FRANÇAISE 1.0

Traduction et mise en forme de *Application Modernization Guide*, édition 1.0, octobre 2026. Les exemples illustratifs et les limites énoncées dans l'original sont conservés.

Table des matières

Note éditoriale	i
1 La modernisation commence par le sens	1
1.1 Trois couches d'une application existante	1
1.2 Pourquoi réécrire le code ne suffit pas	1
1.3 Le contrat de modernisation	1
2 Comprendre la connaissance métier structurée	2
2.1 Passer des libellés à des définitions gouvernées	2
2.2 Distinguer faits, règles et décisions	2
2.3 Formaliser ne signifie pas avoir raison	2
3 Explorer l'application existante	3
3.1 Construire une cartographie des preuves	3
3.2 Une échelle de preuve	3
3.3 Instruments pratiques d'exploration	3
4 Cas fil rouge : confirmer une commande client	4
4.1 Fragment de code historique	4
4.2 L'écart décisif : présente ou valide ?	4
4.3 Enregistrer BR-ORD-014	4
4.4 Classer plutôt que recopier	5
5 Transformer une hypothèse en règle métier acceptée	6
5.1 Trancher à partir de scénarios concrets	6
5.2 Exemple de politique approuvée	6
5.3 La traçabilité comme mécanisme de contrôle	6
6 Un modèle indépendant de connaissance métier	7
6.1 Identité, relations et contraintes	7
6.2 Une spécification portable de BR-ORD-014	7
6.3 Connaissance et spécification conventionnelle	8
6.4 Un cycle de vie est un contrat	8
6.5 Opérationnaliser sans dépendre d'une plateforme	8
7 Démontrer le modèle par les tests	9
7.1 Cas d'acceptation issus de la spécification	9
7.2 Tests de caractérisation et tests d'acceptation	9
7.3 Validation différentielle	10
7.4 Critères de mise en production	10
8 Remplacement progressif et coexistence	11
8.1 Choisir une frontière métier limitée	11
8.2 Trois étapes de déploiement	11
8.3 Responsabilité des états et transactions	11
9 Fiabilité des données et migration sémantique	12
9.1 Des données correctes peuvent conserver un sens erroné	12
9.2 Une réconciliation selon deux axes	12
9.3 Une interface concrète entre données et connaissance	12
10 Une IA appuyée sur une connaissance métier explicite	13

10.1 De la recherche à l'interprétation contrôlée	13
10.2 Une demande de confirmation de commande	13
10.3 Mesurer l'ancrage dans les preuves, pas les slogans	13
11 Un pilote de modernisation concret	14
11.1 Équipe et rôles	14
11.2 Séquence indicative	14
11.3 Quand ne pas employer cette approche	14
11.4 Tableau de bord du pilote	14
12 Gouvernance et évolution continue	15
12.1 Préserver l'indépendance de la connaissance vis-à-vis du code	15
12.2 Éviter de créer un second patrimoine historique non documenté	15
12.3 Le résultat distinctif	15
A Modèle de fiche de connaissance métier	16
B Matrice de traçabilité BR-ORD-014	17
C Appliquer la méthode avec logiCells	18
C.1 Le rôle de logiCells	18
C.2 Délimiter une intervention concrète	18
C.3 Contact et prochaine étape	18
C.4 Ce qui doit être démontré avant une adoption	18
D Liste de vérification avant de démarrer	19

La modernisation commence par le sens

1.1 Trois couches d'une application existante

Une application métier comporte une *couche d'implémentation* (code, stockage, interfaces, infrastructure), une *couche d'observation* (transactions, journaux, incidents et résultats réels) et une *couche sémantique* (concepts, obligations, décisions et significations reconnues). Ces couches sont liées, mais ne sont pas interchangeables. Une base de données peut enregistrer un statut de façon parfaitement fiable sans expliquer l'engagement que ce statut crée.

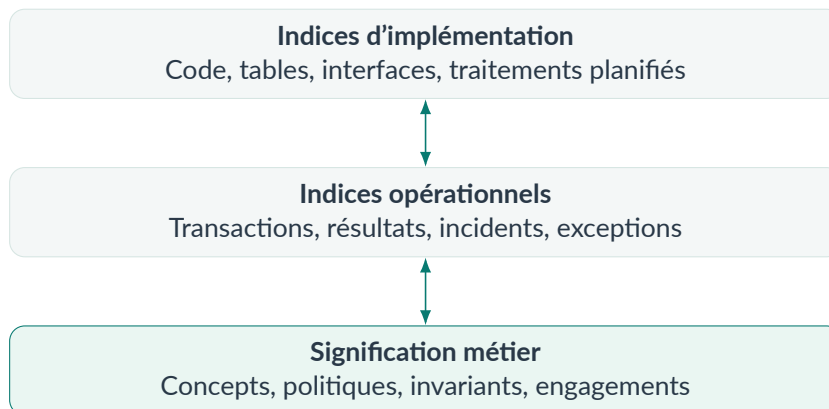


Figure 1.1 – Trois couches à étudier : aucune, prise isolément, n'établit l'intention métier.

1.2 Pourquoi réécrire le code ne suffit pas

Reproduire les détails d'implémentation peut perpétuer des règles obsolètes, des comportements accidentels et des contradictions cachées. À l'inverse, reconcevoir sans s'appuyer sur des preuves peut faire disparaître des exigences que personne ne savait formuler. L'objectif n'est pas d'obtenir un code identique, mais d'assurer une continuité justifiée du sens métier, tout en rendant explicites les changements approuvés.

Principe. Considérer le système historique comme une source d'indices, et non comme la définition indiscutable du métier.

1.3 Le contrat de modernisation

Dès le départ, convenir des décisions critiques à préserver, des anomalies connues à corriger, des responsables habilités à trancher les ambiguïtés et des preuves nécessaires avant de basculer le trafic. Un résultat satisfaisant associe un référentiel de connaissances explicite, une capacité de remplacement testée et une trajectoire maîtrisée vers l'adoption opérationnelle.

Livable : Une charte de modernisation reliant les résultats métier, les risques sémantiques et les critères d'acceptation.

Comprendre la connaissance métier structurée

2.1 Passer des libellés à des définitions gouvernées

Une connaissance structurée ne se réduit pas à un glossaire. Elle attribue une identité aux concepts, définit leurs relations, contraint les états admissibles, identifie les décisions et relie les règles à leurs sources et à leurs tests. Elle est structurée lorsque des participants indépendants peuvent examiner une affirmation, détecter une incohérence et déterminer si deux implémentations la respectent.

Élément	Illustration en gestion des commandes
Concept	Une commande client est une demande susceptible de devenir un engagement commercial accepté.
Relation	Une commande appartient à un client et contient des lignes de commande.
Invariant	Une commande confirmée doit respecter la politique d'éligibilité à la confirmation approuvée.
Transition	La confirmation transforme un brouillon en engagement confirmé.
Décision	Cette commande précise est-elle actuellement éligible à la confirmation ?
Preuve	Parcours du code, trace transactionnelle, politique approuvée et test d'acceptation.

Tableau 2.1 – Composantes d'une connaissance métier structurée.

2.2 Distinguer faits, règles et décisions

Un *fait* décrit une condition observée ou enregistrée. Une *règle* définit ce qui doit être vrai ou peut être autorisé. Une *décision* applique les règles pertinentes à des faits dans un contexte donné. Ces distinctions sont importantes pour l'intelligence artificielle : une explication générée peut évoquer des faits sans pour autant être autorisée à modifier les règles ou à déclencher une action métier.

2.3 Formaliser ne signifie pas avoir raison

Un diagramme ou une expression de règle rend les hypothèses examinables, mais ne les rend pas vraies. Le responsable métier doit trancher, par exemple, si une « ligne valide » suppose un produit actif, une quantité positive, un prix autorisé ou une allocation de stock réussie. Les questions ouvertes doivent être consignées comme des artefacts à part entière plutôt que résolues par supposition.

Livrable : Un glossaire du domaine, un registre des politiques, une cartographie des cycles de vie et une liste des questions non résolues.

Explorer l'application existante

3.1 Construire une cartographie des preuves

Commencer par délimiter les capacités, les bases de données, les appels de services, les traitements par lots, les rapports opérationnels, les rôles utilisateurs et les traitements d'exception. Recueillir des traces représentatives de transactions ainsi que la configuration de déploiement. Relier chaque règle proposée aux artefacts qui la justifient, avec leurs identifiants de version et les dates d'observation.

3.2 Une échelle de preuve

1. **Candidate** : déduite d'un artefact par un analyste ou une IA.
2. **Observée** : reproduite à partir d'une analyse du code ou de preuves d'exécution.
3. **Corroborée** : étayée par des sources indépendantes ou par des parties prenantes.
4. **Approuvée** : reconnue comme politique intentionnelle par un responsable métier habilité.
5. **Vérifiée** : couverte par des tests d'acceptation exécutables sur la cible.

Cette échelle suit différentes formes de confiance. Elle ne signifie pas qu'un test exécuté uniquement sur le code historique démontre l'intention métier. L'IA peut accélérer l'indexation et la formulation d'hypothèses, mais elle ne peut transformer une inférence non examinée en règle métier approuvée.

3.3 Instruments pratiques d'exploration

L'analyse statique fait apparaître les branchements, les validations, les écritures en base et les dépendances. Le profilage des bases de données révèle les distributions et les valeurs atypiques. Les journaux, traces et historiques d'incidents éclairent les parcours exceptionnels. La fouille de processus peut mettre en évidence les variantes de workflow réellement suivies, lorsque les événements sont de qualité suffisante. Les entretiens permettent de clarifier l'intention et les conséquences métier. Utiliser des données synthétiques ou convenablement protégées, et consigner les limites des éléments recueillis.

Livable : Une cartographie du système, un registre des preuves et un catalogue priorisé des règles candidates.

Cas fil rouge : confirmer une commande client

4.1 Fragment de code historique

Considérons l'implémentation illustrative suivante :

```
if Order.Status = Draft then
  if Order.LineCount > 0 then
    if Customer.IsActive then begin
      Order.Status := Confirmed;
      ReserveStock(Order);
    end;
```

Exemple 4.1 – Parcours historique illustratif de confirmation

Ce fragment laisse supposer des vérifications d'éligibilité et un effet secondaire. Il ne permet pas de savoir si un échec de réservation annule la confirmation, si les requêtes répétées sont sûres ou si une commande non vide contient réellement des lignes utilisables.

4.2 L'écart décisif : présente ou valide ?

Le code teste `LineCount > 0`. L'énoncé métier proposé pendant l'exploration est : « une commande ne peut être confirmée que si elle contient au moins une ligne valide ». Ces deux conditions sont différentes. Une commande comprenant une unique ligne annulée satisfait le test historique, mais peut violer la politique métier proposée. Cet écart n'est pas un détail d'implémentation : il représente précisément la connaissance manquante que la modernisation doit mettre au jour.

Principe. Ne jamais remplacer silencieusement un comportement historique observé par une règle métier plus exigeante. Consigner l'écart, obtenir une décision et tester les deux comportements.

4.3 Enregistrer BR-ORD-014

Champ	Entrée de travail examinée
Identifiant	BR-ORD-014
Question métier	À quel moment une commande client à l'état brouillon peut-elle devenir un engagement confirmé ?
Observation historique	Une commande brouillon passe la validation si elle contient plus de zéro ligne et si son client est actif.
Règle cible proposée	Une commande brouillon peut être confirmée si le client est éligible et si au moins une ligne satisfait la politique approuvée de validité des lignes.

Champ	Entrée de travail examinée (suite)
Définitions ouvertes	Client éligible ; ligne valide ; traitement des lignes exclues ; atomi- cité de réservation ; soumissions en double.
Responsable	Responsable de la politique de gestion des commandes (à dési- gner).
Niveau de preuve	<i>Observée</i> pour la condition historique ; <i>Candidate</i> pour la poli- tique proposée (assertions de preuve distinctes).
Statut de la politique	Brouillon / en attente d'approbation.
Classification	Comportement historique déprécié (classement provisoire) ; correction cible dans l'attente de l'approbation métier.
Preuves	Emplacement du code source, trace représentative, décision d'atelier et identifiants des tests (à joindre).

4.4 Classer plutôt que recopier

Pour chaque comportement découvert, choisir l'une des quatre dispositions : *requis*, *toléré*, *déprécié* ou *défaut*. Un comportement toléré peut n'être pris en charge que pendant la migration ; un défaut ne doit pas être réimplémenté simplement parce qu'il est observable. Enregistrer chaque classification, sa justification et son approbateur.

Transformer une hypothèse en règle métier acceptée

5.1 Trancher à partir de scénarios concrets

L'atelier métier doit répondre à des questions illustrées par des cas : commande vide ; ligne unique annulée ; ligne active de quantité nulle ; client inactif ; stock insuffisant ; seconde demande de confirmation ; annulation concurrente. L'atelier doit produire une table de décision, et non de simples comptes rendus de réunion.

Scénario	Décision cible	Justification
Brouillon ; client actif ; ligne valide	Autoriser	Conditions de la politique satisfaites
Brouillon ; client actif ; uniquement des lignes invalides	Refuser	Absence de ligne valide
Brouillon ; client inactif ; ligne valide	Refuser	Client non éligible
Commande déjà confirmée ; requête répétée	Politique explicite	Décision d'idempotence nécessaire
Brouillon ; ligne valide ; aucun stock	Refuser	Dans la politique cible illustrative, l'échec de la réservation interdit la confirmation
Brouillon ; une ligne valide et une ligne annulée	Politique explicite	Définir l'exclusion, la conservation et le reporting
Brouillon ; réservation partielle réussie	Compenser ou annuler atomiquement	Aucun stock ne doit rester réservé après l'échec

Tableau 5.1 – Scénarios de décision pour la confirmation d'une commande.

5.2 Exemple de politique approuvée

À titre purement illustratif, supposons que le métier approuve les définitions suivantes : une ligne valide comporte un produit commercialisable, une quantité strictement positive et un statut non annulé ; un client éligible doit être actif. Toutes les lignes restent dans la commande pour la traçabilité, mais seules les lignes valides entrent dans la réservation de stock. Les lignes annulées, les lignes de quantité nulle et celles portant sur un produit non commercialisable en sont exclues : ce choix explicite exige une approbation métier.

La réservation s'effectue en tout ou rien : toute immobilisation partielle est annulée, la commande restant à l'état **Draft**. Une requête répétée avec la même clé d'idempotence ne doit ni réserver le stock deux fois ni émettre une seconde confirmation. La confirmation ne réussit que si la réservation réussit. Ce sont des décisions illustratives, et non des faits déduits du fragment de code précédent.

5.3 La traçabilité comme mécanisme de contrôle

Maintenir des liens entre observation, hypothèse, décision métier, élément de modèle, implémentation et test d'acceptation. Lorsqu'un lien change, identifier les cas et parties prenantes concernés. L'équivalence sémantique est un argument vérifiable reposant sur cette chaîne ; elle ne se présume pas à partir du seul nombre de lignes correspondant.

Livable : Une table de décision approuvée, un registre des dispositions et une version traçable de la politique.

Un modèle indépendant de connaissance métier

6.1 Identité, relations et contraintes

Modéliser `SalesOrder`, `Customer`, `OrderLine` et `Product` comme des concepts, sans les recopier directement depuis les tables de base de données. Définir les identités et cycles de vie indépendamment des clés physiques. Énoncer les relations (la commande possède des lignes ; chaque ligne référence un produit ; la commande est associée à un client) et documenter explicitement la cardinalité, la propriété et l'interprétation temporelle.

6.2 Une spécification portable de BR-ORD-014

Le pseudocode suivant décrit une spécification indépendante des plateformes. Il n'est *pas* présenté comme un langage spécifique au domaine (DSL) directement exécutable.

```
ValidLine(line) :=
  line.quantity > 0
  AND line.product.isSellable
  AND line.status != Cancelled

CanConfirm(order) :=
  order.state = Draft
  AND order.customer.isActive
  AND EXISTS line IN order.lines: ValidLine(line)

Confirm(order):
  IF completed result exists for request key:
    RETURN previous result without new side effects
  PRECONDITION CanConfirm(order)
  ELIGIBLE_LINES := FILTER order.lines BY ValidLine
  EXCLUDE all invalid lines from reservation
  TRY atomic reservation of ELIGIBLE_LINES:
    IF all reservations committed:
      TRANSITION order Draft -> Confirmed
      EMIT OrderConfirmed once
    ELSE:
      RELEASE any partial reservations
      KEEP state Draft; record failure
```

Exemple 6.1 – Spécification de politique indépendante des technologies

La clé de requête est vérifiée avant l'éligibilité dépendant de l'état ; autrement, une relance après confirmation échouerait sur la précondition `Draft`. Les demandes concurrentes exigent également un enregistrement d'idempotence durable et atomique, avec gestion sûre du cas où la première demande est encore en cours. Une transaction de base de données, un service de réservation ou une saga avec compensations doit garantir le résultat approuvé « tout ou rien ». L'échec d'une réservation n'est pas l'annulation d'une confirmation déjà terminée : dans ce cas, la commande ne doit jamais atteindre l'état `Confirmed`. Si certaines réservations réussissent avant

qu'une autre échoue, une libération compensatoire ou un protocole atomique est indispensable. L'ordre entre réservation de stock, modification d'état et publication d'événement exige une conception transactionnelle. La spécification consigne le contrat observable visé sans imposer les mécanismes d'implémentation.

6.3 Connaissance et spécification conventionnelle

Une spécification fonctionnelle classique décrit généralement une fonctionnalité ou un écran à un moment donné. Un modèle de connaissance gouverné relie identités métier, définitions, conditions d'éligibilité, changements d'état, responsabilités, preuves, versions de politiques et vérification exécutable au travers des implémentations. Une bonne spécification peut contenir tous ces éléments : la différence ne tient pas au format du document, mais au fait que le sens soit explicite, maintenu indépendamment, traçable et réutilisable. Dans BR-ORD-014, la même définition de **ValidLine** sert aux décisions de confirmation, aux explications fournies à l'utilisateur, à la conception des tests et à l'analyse d'impact d'une évolution de politique.

6.4 Un cycle de vie est un contrat

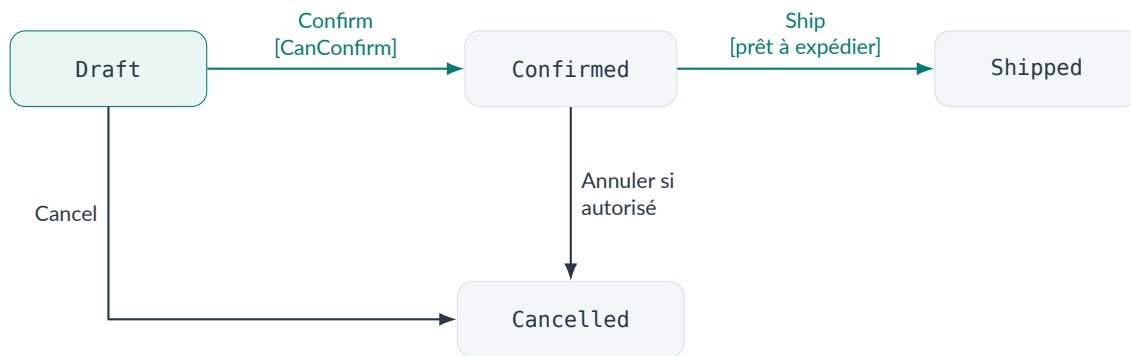


Figure 6.1 – Cycle de vie illustratif d'une commande : les conditions de passage doivent être décidées et testées.

6.5 Opérationnaliser sans dépendre d'une plateforme

Le modèle de connaissance devient utile lorsqu'il alimente plusieurs artefacts contrôlés : service de décision, messages d'éligibilité dans l'interface, tests de validation, documentation et explications assistées par IA. Ces usages doivent partager une sémantique gouvernée tout en conservant leurs propres frontières de sécurité et de déploiement. Un modèle n'a pas besoin d'être lui-même un moteur exécutable pour avoir de la valeur ; si l'exécution est nécessaire, son implémentation doit être testée au regard du contrat du modèle.

Livrable : Un modèle de concepts versionné, une politique formalisée, un contrat de transition et les correspondances avec les implémentations.

Démontrer le modèle par les tests

7.1 Cas d'acceptation issus de la spécification

Les scénarios suivants traduisent en vérifications concrètes le contrat cible présenté au chapitre précédent. Les mots-clés **GIVEN**, **WHEN** et **THEN** conservent leur rôle de notation de test.

```
AC-014-01
GIVEN Draft order, active customer, one valid line
WHEN Confirm is requested
THEN state becomes Confirmed; confirmation event recorded

AC-014-02
GIVEN Draft order, active customer, one cancelled line
WHEN Confirm is requested
THEN confirmation rejected; state remains Draft

AC-014-03
GIVEN Draft order, inactive customer, one valid line
WHEN Confirm is requested
THEN confirmation rejected; state remains Draft

AC-014-04
GIVEN Draft order, active customer, one valid line
AND stock reservation fails
WHEN Confirm is requested
THEN state remains Draft; any partial holds are released
AND failure is auditable; no confirmation event emitted

AC-014-05
GIVEN Draft order, one valid and one cancelled line
WHEN Confirm is requested
THEN reserve only the valid line; retain both lines
AND record exclusion in decision audit

AC-014-06
GIVEN a previously successful confirmation with request key K
WHEN the same confirmation request K is retried
THEN return prior result; reserve no stock again
AND emit no duplicate OrderConfirmed event
```

Exemple 7.1 – Cas d'acceptation illustratifs au niveau de la spécification

7.2 Tests de caractérisation et tests d'acceptation

Un test de caractérisation documente ce que fait le système historique. Un test d'acceptation vérifie le comportement cible approuvé. Dans le scénario d'une ligne annulée, leurs résultats peuvent volontairement diverger : cet écart correspond à une correction approuvée et non automatiquement à une régression. Conserver les deux ensembles de tests distincts.

7.3 Validation différentielle

Exécuter des cas comparables sur les parcours ancien et nouveau dans des environnements isolés. Comparer décisions, changements d'état, effets en aval, idempotence et traces d'audit. Classer chaque différence comme amélioration connue, tolérance approuvée ou régression non résolue. Des nombres d'enregistrements et des totaux identiques ne démontrent pas à eux seuls la préservation du sens métier.

7.4 Critères de mise en production

Exiger la validation formelle des décisions de politique à fort impact, la réussite des tests d'acceptation cibles, l'examen des traitements d'exception, des répétitions de la procédure de retour arrière, les tests de compensation des réservations et la mise en place de la supervision. Les cas à haut risque touchant à l'argent, aux contrats ou aux obligations réglementaires nécessitent des contrôles proportionnés.

Livable : Un catalogue de cas exécutables, un registre des écarts et des éléments de preuve justifiant l'autorisation de mise en production.

Remplacement progressif et coexistence

8.1 Choisir une frontière métier limitée

Commencer par une décision clairement délimitée, aux résultats observables, relevant de responsables métier identifiés. Pour le cas des commandes, la décision d'éligibilité peut constituer un meilleur pilote que le remplacement de l'ensemble du module de traitement des commandes. Les applications existantes peuvent continuer à être maîtres des données pendant l'évaluation du nouveau parcours de décision.

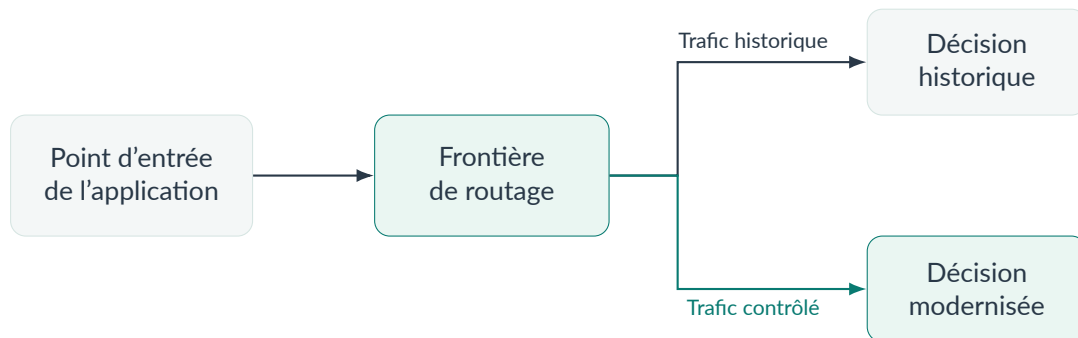


Figure 8.1 – Une frontière contrôlée permet une évaluation en parallèle et un routage partiel.

8.2 Trois étapes de déploiement

1. **Mode fantôme (shadow)** : calculer la décision modernisée sans effet opérationnel et collecter les écarts expliqués.
2. **Routage partiel** : orienter une petite population éligible vers la nouvelle décision, sous le contrôle de mécanismes d'activation de fonctionnalités.
3. **Retrait** : supprimer l'ancien parcours une fois les preuves, les vérifications de dépendances et les critères de retour arrière satisfaits.

8.3 Responsabilité des états et transactions

Désigner, pour chaque état, un composant faisant autorité pour l'écriture et définir les contrats d'intégration. Éviter les doubles écritures sans réconciliation. Rendre explicites l'ordre des événements, les nouvelles tentatives, les livraisons en double et la cohérence éventuelle : cette dernière devient une question de politique métier dès lors qu'elle affecte les engagements, les stocks ou l'argent.

Livrable : Un plan de bascule, des contrôles de routage, une matrice de responsabilités et une procédure de retour arrière.

Fiabilité des données et migration sémantique

9.1 Des données correctes peuvent conserver un sens erroné

Les contrôles de qualité établissent si les faits sont complets, cohérents, actuels et traçables. Les contrôles sémantiques établissent ce que signifient ces faits. Un code de statut peut être parfaitement renseigné tout en confondant « approuvé », « réservé » et « engagé ». Pour chaque champ migré, consigner son interprétation dans le domaine et les hypothèses retenues pour toute transformation.

9.2 Une réconciliation selon deux axes

Vérifier la réconciliation quantitative (enregistrements, soldes, couverture des clés) et la réconciliation sémantique (états, obligations, résultats des règles, signification temporelle). Ces deux axes sont complémentaires. Une migration réussie doit également conserver la filiation des données et rendre explicite toute perte de précision.

9.3 Une interface concrète entre données et connaissance

Dans le cas de la confirmation de commande, une source soumise à des contrôles qualité peut fournir `customer.active`, la possibilité de vendre le produit, les quantités des lignes et leur statut d'annulation. Le modèle métier définit comment ces faits entrent dans `CanConfirm`. Le résultat de la décision doit exposer un code de motif et référencer la version de politique appliquée. La fiabilité rend les données d'entrée dignes de confiance ; la sémantique rend leur emploi intelligible.

Livrable : Une correspondance entre champs et concepts, des rapports de réconciliation et un contrat de décision versionné par politique.

Une IA appuyée sur une connaissance métier explicite

10.1 De la recherche à l'interprétation contrôlée

Un assistant d'intelligence artificielle peut retrouver des éléments de preuve et résumer des règles, mais les données retrouvées ne confèrent aucune autorité pour modifier une politique. Un assistant ancré dans la connaissance doit identifier la version de politique en vigueur, expliquer comment les faits satisfont ou non les conditions et signaler les preuves manquantes ou contradictoires. L'exécution d'actions doit rester soumise à des frontières explicites d'autorisation et de validation.

10.2 Une demande de confirmation de commande

Un utilisateur demande : « Peut-on confirmer la commande 482 ? » Le système récupère l'état, l'éligibilité du client et les lignes, évalue la politique approuvée par un parcours de décision digne de confiance, puis restitue l'éligibilité, les motifs, les incertitudes et les références. L'assistant ne déduit pas la validité d'une ligne de la simple présence d'un enregistrement et ne doit pas engager la commande au seul motif qu'il a formulé une explication plausible.

10.3 Mesurer l'ancrage dans les preuves, pas les slogans

Mesurer la proportion des réponses étayées par des versions de politiques citées, la concordance avec les décisions testées, les ambiguïtés non résolues et les tentatives d'action dangereuses bloquées. Ne pas confondre une étiquette comme « prêt pour l'IA » avec un indicateur de réussite opérationnelle.

Un pilote de modernisation concret

11.1 Équipe et rôles

Un pilote pragmatique associe un responsable de politique métier, un spécialiste du système historique, un analyste du domaine, un architecte de solution, un ingénieur ou ingénieur de test, ainsi qu'un représentant de l'exploitation ou de la sécurité. Dans une petite organisation, une même personne peut cumuler plusieurs rôles ; le pouvoir d'approbation et la responsabilité d'implémentation doivent cependant rester distinguables.

11.2 Séquence indicative

Un pilote de six à dix semaines constitue une hypothèse de planification, *pas* une promesse de livraison. Semaines 1 à 2 : inventaire et collecte des preuves. Semaines 3 à 4 : ateliers, modèle et cas approuvés. Semaines 5 à 7 : prototype de parcours de décision et tests différentiels. Semaines 8 à 10 : évaluation du déploiement contrôlé, revue et décision de poursuivre ou non (*go/no-go*). Des intégrations complexes ou des preuves insuffisantes peuvent allonger sensiblement ce calendrier.

11.3 Quand ne pas employer cette approche

Éviter une initiative fortement axée sur l'extraction de connaissances lorsque l'application sera retirée sans remplacement, lorsque la capacité métier est triviale ou standardisée, lorsqu'aucun expert responsable du domaine ne peut valider la sémantique retrouvée ou lorsqu'un remplacement par progiciel à prix fixe constitue clairement une meilleure solution qu'une reconstruction spécifique. En présence de risques élevés pour la sûreté ou la réglementation, des garanties et des vérifications formelles supplémentaires peuvent être nécessaires.

11.4 Tableau de bord du pilote

Dimension	Preuve de progression
Couverture	Décisions critiques découvertes, classées et attribuées à un responsable.
Traçabilité	Règles approuvées reliées à des preuves et à des cas d'acceptation.
Qualité sémantique	Écarts résolus ou acceptés en connaissance de cause.
Exploitabilité	Comportement mesuré, retour arrière et observabilité.
Transférabilité	Modèle réutilisé dans la documentation, les décisions et les tests.

Tableau 11.1 – Indicateurs de progression d'un pilote de modernisation.

Livrable : Un dossier de pilote présentant des résultats mesurables et une décision explicite de poursuite ou d'arrêt.

Gouvernance et évolution continue

12.1 Préserver l'indépendance de la connaissance vis-à-vis du code

Attribuer aux règles métier des identifiants de version, des responsables et des processus de revue. Relier chaque règle aux applications et aux tests concernés. Une modification de politique doit déclencher une analyse d'impact avant le déploiement, et les incidents opérationnels doivent enrichir les preuves associées au modèle.

12.2 Éviter de créer un second patrimoine historique non documenté

Un modèle devient lui-même un artefact historique s'il dérive par rapport à l'exécution. Automatiser autant que possible les vérifications et comparer périodiquement modèle, implémentation courante et résultats observés. Consigner les exceptions plutôt que les enfouir dans des correctifs locaux.

12.3 Le résultat distinctif

La modernisation crée une valeur durable lorsque l'organisation peut discuter d'une décision métier et la modifier sans devoir commencer par déconstruire son implémentation. Le résultat réutilisable est une compréhension gouvernée des comportements métier, étayée par des preuves, des tests et des contrats opérationnels explicites.

Modèle de fiche de connaissance métier

Champ	Contenu attendu
Identifiant et titre	Identifiant stable ; assertion métier concise.
Périmètre	Capacité, entités, canaux et date d'effet.
Preuves	Emplacement de la source, version, traces et références d'entretiens.
Niveau de preuve	Candidate / Observée / Corroborée / Approuvée / Vérifiée ; statut propre à l'assertion considérée.
Observation	Ce que fait effectivement et de façon démontrable l'implémentation.
Politique visée	Énoncé approuvé par le métier et exemples associés.
Ambiguïtés	Définitions non résolues et cas limites.
Disposition	Requis / toléré / déprécié / défaut.
Responsable et cycle de vie	Approbateur habilité ; brouillon / approuvé / remplacé.
Traçabilité	Section du modèle, contrat d'implémentation et tests.

Matrice de traçabilité BR-ORD-014

ID	Observation ou politique	Représentation explicite	Vérification
BR-ORD-014	Brouillon, client actif, lignes non vides dans le système historique	CanConfirm avec ValidLine, selon la cible approuvée	AC-014-01 : accepter une ligne valide
BR-ORD-014-A	Le système historique accepte une seule ligne, même annulée	La cible exclut les lignes annulées ; différence de comportement approuvée	AC-014-02 : refuser une commande sans ligne valide
BR-ORD-014-B	Vérification du statut actif du client	L'éligibilité exige un client actif	AC-014-03 : refuser un client inactif
BR-ORD-014-C	Échec et succès partiel de réservation non spécifiés	Réservation en tout ou rien ; libérer les réservations partielles avant confirmation	AC-014-04 : compensation partielle
BR-ORD-014-D	Présence de lignes valides et annulées	Conserver toutes les lignes ; ne réserver que les valides (illustratif)	AC-014-05 : lignes mixtes
BR-ORD-014-E	Requêtes répétées lors de nouvelles tentatives	Clé d'idempotence restituant le résultat précédent, sans répéter les effets	AC-014-06 : requêtes en double

Appliquer la méthode avec logiCells

C.1 Le rôle de logiCells

logiCells promeut la modernisation des applications par la récupération, la structuration et la gouvernance de la connaissance métier. Ce guide présente volontairement la méthode sans imposer de grammaire propriétaire ni de runtime particulier. Au-delà d'un simple exercice de documentation, logiCells positionne la connaissance métier structurée et la modélisation conceptuelle comme des actifs susceptibles d'éclairer le comportement des applications et leur évolution ; son orientation architecturale publiée comprend notamment des modèles conceptuels et un hypergraphe de connaissances.

Le périmètre réel de l'outillage et de l'automatisation doit être vérifié au cours d'un pilote. Une intervention logiCells peut s'appuyer sur ce cadre pour délimiter un périmètre applicatif, recueillir les preuves, organiser les ateliers de domaine, maintenir la traçabilité des règles et planifier un remplacement progressif. Il s'agit d'activités d'accompagnement proposées, et non de l'affirmation qu'un produit exécuterait automatiquement chacune de ces étapes.

C.2 Délimiter une intervention concrète

Le meilleur point de départ est une décision métier limitée mais importante, comme BR-ORD-014. L'équipe conjointe identifie les sources faisant autorité, rend visibles les ambiguïtés, obtient l'approbation métier et établit un contrat testable avant de sélectionner une technologie d'implémentation. Le livrable est un référentiel de connaissance versionné et examinable, accompagné de cas d'acceptation et d'un plan d'intégration. Les choix d'architecture et d'outillage demeurent des décisions explicites, et non des préalables à la compréhension de la méthode.

C.3 Contact et prochaine étape

Pour une démonstration délimitée, du type BR-ORD-014, consulter <https://www.logicells.com/> ou écrire à contact@logicells.com. Demander un pilote documenté reliant une décision métier, depuis les indices historiques jusqu'à l'approbation, au modèle, aux tests et au déploiement contrôlé.

C.4 Ce qui doit être démontré avant une adoption

Exiger un pilote complet présentant une trace du système historique, une table de décision approuvée, un modèle de connaissance maintenu, des tests d'acceptation et de caractérisation, la traçabilité des impacts d'un changement et un déploiement mesuré. Exiger des preuves pour toute affirmation relative aux capacités d'un outil : cette annexe ne formule délibérément aucune affirmation non vérifiée sur l'analyse automatisée de code, la génération de code ou l'exécution.

Liste de vérification avant de démarrer

- ☐ Chaque assertion métier importante possède-t-elle un responsable, des preuves et une version ?
- ☐ Les observations historiques sont-elles explicitement distinguées des décisions cibles ?
- ☐ Les scénarios limites ont-ils été examinés, notamment les demandes en double et les échecs ?
- ☐ Le modèle de connaissance et le contrat opérationnel sont-ils compréhensibles indépendamment d'un fournisseur donné ?
- ☐ Les tests sont-ils reliés aux décisions approuvées, et pas seulement aux anciens comportements observés ?
- ☐ Les limites techniques, les incertitudes et les risques de déploiement sont-ils documentés ?
- ☐ Les diagrammes, le glossaire et les exemples ont-ils été revus par un représentant métier et un représentant technique ?