

## 01 • Diagnosi

# Non vi manca un'app. Vi manca qualcuno che risponda del software.

Il software diventa un problema quando il processo dipende da fogli, integrazioni fragili o un fornitore che conosce tutto ma documenta poco. Prima di costruire, serve capire cosa deve reggere davvero.

## Segnali da mettere a fuoco

- Un processo critico passa tra Excel, email e inserimenti manuali.
- Il sistema esistente rallenta nuove offerte, canali o modalità operative.
- Ogni modifica richiede la disponibilità di una persona o di un fornitore specifico.
- Integrazioni e dati sono trattati come eccezioni, non come parte del prodotto.
- Il go-live è considerato la fine del lavoro, senza un piano di evoluzione.

*Software in Mano significa un percorso unico dal problema al rilascio e all'evoluzione, con perimetro, responsabilità e proprietà del codice chiari.*

### Domanda diagnostica

Se il sistema si fermasse domani, quale processo del business tornerebbe manuale — e chi saprebbe rimetterlo in funzione?

## 02 · Costo dell'inazione

# Il costo cresce mentre il problema sembra ancora gestibile.

Il lavoro manuale e il software fragile spesso non compaiono come una singola voce di costo. Si distribuiscono in ritardi, errori, dipendenze e opportunità rimandate.

**Tempo operativo**

Persone qualificate copiano, riconciliano o verificano dati invece di gestire eccezioni e clienti.

**Errori silenziosi**

Passaggi manuali e fonti duplicate producono differenze scoperte tardi.

**Dipendenza**

Conoscenza, accessi e decisioni tecniche restano fuori dal controllo dell'azienda.

**Crescita bloccata**

Nuovi volumi, canali o integrazioni richiedono workaround invece di capacità riutilizzabile.

## Costruite una baseline prima del preventivo

- Mappate attori, passaggi, sistemi e punti in cui il dato viene ricopiato.
- Separate problemi frequenti da eventi rari ma ad alto impatto.
- Identificate chi decide, chi usa il sistema e chi ne gestisce le eccezioni.
- Definite cosa deve essere vero per considerare riuscito il primo rilascio.

*Un preventivo senza una baseline non elimina l'incertezza: la sposta nel progetto.*

## 03 · Metodo

# Discovery, progetto, evoluzione: una sequenza, non tre promesse.

Ogni fase produce un risultato controllabile e un gate di decisione. Il cliente può procedere, ridurre il perimetro o fermarsi con documenti utilizzabili.

|                       |                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------|
| 1. Discovery          | Problema, utenti, flussi, integrazioni, rischi, architettura e criteri di accettazione. |
| 2. Decisione          | Scope incluso ed escluso, milestone, stima e rischi espliciti prima del build.          |
| 3. Progetto           | Consegne verificabili, review, test e dimostrazioni sul flusso reale.                   |
| 4. Messa in esercizio | Deploy, accessi, documentazione, formazione e gestione del passaggio.                   |
| 5. Retainer evolutivo | Priorità e ore mensili chiare per migliorare ciò che è in produzione.                   |

## AI come metodo, non come etichetta

Una funzione AI entra solo quando il processo, i dati, il rischio di errore e il criterio di successo sono chiari. Deve esistere un fallback e una responsabilità umana.

## 04 · Output

# Cosa ricevete prima, durante e dopo il codice.

Gli output esatti dipendono dal contesto. Questi esempi rendono concreto il livello di controllo atteso.

**Mappa del flusso**

Attori, dati, passaggi, eccezioni e sistemi coinvolti.

**Specifica verificabile**

Comportamenti e criteri di accettazione comprensibili al business.

**Architettura motivata**

Scelte, integrazioni e trade-off documentati senza tecnologie decorative.

**Piano di rilascio**

Milestone, dipendenze, responsabilità e condizioni di go/no-go.

**Quality gate**

Build, test, sicurezza di base, osservabilità e controllo dello scope.

**Handoff reale**

Repo, accessi, documentazione e conoscenza restano disponibili al cliente.

## Principi di controllo

- Il codice e il repository sono del cliente.
- Lo scope fuori perimetro richiede una decisione esplicita, non una sorpresa a consuntivo.
- Le integrazioni critiche vengono trattate come boundary da progettare e verificare.
- Dopo il rilascio esiste un percorso di gestione, evoluzione o handoff.

## 05 · Fit e prossimo passo

# Pagate prima per ridurre l'incertezza, non per nascondere la nel progetto.

La Discovery è un prodotto autonomo: produce scope, architettura e stima decisionale. Se non si prosegue, il materiale resta utilizzabile.

**È un buon fit se**

- Dovete digitalizzare un processo o recuperare software che non regge.
- Il sistema deve integrarsi con dati e strumenti già presenti.
- Volete proprietà del codice e una continuità dopo il go-live.
- Siete disponibili a coinvolgere utenti e decisore nella Discovery.

**Non è il fit giusto se**

- Cercate solo sviluppatori a giornata o il prezzo/ora più basso.
- Volete un preventivo fisso senza chiarire problema, utenti e rischi.
- La deadline è già decisa ma nessuno può definire cosa significa accettare il lavoro.
- Avete già un team capace e vi manca soltanto la guida tecnica.

## Avvia una Discovery a pagamento

Primo passo: contesto, flussi, scope, architettura, rischi e stima. Un documento decisionale prima di impegnare budget sul build.

[108vision.it/software-in-mano](https://108vision.it/software-in-mano)