

marksync

Contract-Based AI-Human-Algorithm Conversation
for Live Process Deployment & Control
with Self-Learning

1. Prompt -> System

Jedno zdanie uruchamia cały proces

INPUT: PROMPT UZYTKOWNIKA

***Build an order management
REST API with human approval
before payment***

OCZEKIWANIE: CO SYSTEM POWINIEN ZROBIC

System powinien:

1. Sparsowac intencje z promptu
2. Zidentyfikowac typ uslugi: REST API
3. Wykryc wymaganie: human approval
4. Wygenerowac README.md z blokami markpact:*
5. Ustawic pipeline: script + llm + human

WALIDACJA

- [v] Prompt sparsowany
IntentParser.parse() -> service_type=api
- [v] Human-in-the-loop wykryty
'human approval' -> actor: human
- [v] Kontrakt wygenerowany
README.md z 7 blokami markpact:*

2. Kontrakt README.md -- utworzony

Markdown = zrodlo prawdy dla calego procesu

README.MD -- AKTUALNY STAN

Order Management API

REST API z human approval przed platnoscia.

```
```yaml markpact:orchestration
```

```
pipeline:
 name: order-api
 steps:
 - name: parse_order
 actor: script
 - name: check_inventory
 actor: llm
 - name: fraud_detection
 actor: script
 - name: human_payment_approval
 actor: human
 config: {channel: web, timeout: 7200}
 - name: process_payment
 actor: script
 - name: deploy_update
 actor: llm
  ```
```

```
```json markpact:deps
```

```
{ "runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"] }
```

```
```python markpact:file=app/main.py
```

```
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
    return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
    return {"status": "ok"}
  ```
```

```
```bash markpact:run
```

```
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

```
```yaml markpact:deploy
```

```
target: docker
pactown:
 name: order-api
 services:
 api: {port: 8000, health_check: /health}
  ```
```

```
```json markpact:state
```

```
{ "phase": "init", "health": null, "success_count": 0, "error_count": 0 }
```

```
```text markpact:log
```

```
[T+0s] CONTRACT_CREATED
```

OCZEKIWANIE: WYMAGANE BLOKI

README.md musi zawierac:

- [V] markpact:orchestration -- pipeline z krokami
- [V] markpact:deps -- zaleznosci runtime
- [V] markpact:file=app/main.py -- kod aplikacji
- [V] markpact:run -- komenda uruchomienia
- [V] markpact:deploy -- konfiguracja deployu
- [V] markpact:state -- stan procesu (phase: init)
- [V] markpact:log -- historia zdarzen

Walidacja: 7/7 blokow obecnych [V]

WALIDACJA

- [V] BlockParser.parse() -> 7 blokow
orchestration, deps, file, run, deploy, state, log
- [V] SHA-256 hash kazdego bloku
Integralnosc zweryfikowana
- [V] Pipeline ma 6 krokow
3x script + 2x llm + 1x human
- [V] markpact:state.phase == init
Stan poczatkowy OK

3. Pipeline Start

PipelineEngine czyta markpact:orchestration

README.MD -- MARKPACT:STATE ZMIENIONY

```
# Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
 name: order-api
 steps:
 - name: parse_order
 actor: script
 - name: check_inventory
 actor: llm
 - name: fraud_detection
 actor: script
 - name: human_payment_approval
 actor: human
 config: {channel: web, timeout: 7200}
 - name: process_payment
 actor: script
 - name: deploy_update
 actor: llm
 ...

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
 return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
 return {"status": "ok"}
```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```yaml markpact:deploy
target: docker
pactown:
 name: order-api
 services:
 api: {port: 8000, health_check: /health}
```json markpact:state
{"phase": "running", "health": null, "success_count": 0, "error_count": 0}
```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

## OCZEKIWANIE: ENGINE URUCHOMIONY

Co sie dzieje:

1. PipelineEngine parsuje markpact:orchestration
2. Tworzy PipelineRun z 6 krokami
3. markpact:state: init -> running
4. markpact:log otrzymuje wpis

[phase: init] --> [phase: running]

## WALIDACJA

- [V] PipelineRun utworzony  
run\_id=run-001, 6 steps
- [V] markpact:state zaktualizowany  
phase: init -> running
- [V] markpact:log wpis dodany  
PIPELINE\_STARTED: order-api

## 4. Step: parse\_order

actor: script -- Walidacja schematu

### README.MD -- MARKPACT:LOG +1

```
Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
  name: order-api
  steps:
    - name: parse_order
      actor: script
    - name: check_inventory
      actor: llm
    - name: fraud_detection
      actor: script
    - name: human_payment_approval
      actor: human
      config: {channel: web, timeout: 7200}
    - name: process_payment
      actor: script
    - name: deploy_update
      actor: llm
  ...

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
    return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
    return {"status": "ok"}
```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```yaml markpact:deploy
target: docker
pactown:
  name: order-api
  services:
    api: {port: 8000, health_check: /health}
```json markpact:state
{"phase": "running", "health": null, "success_count": 0, "error_count": 0}
```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

OCZEKIWANIE: SKRYPT WALIDACYJNY

[ACTOR: script] Deterministyczny kod

```
ten sam input = ten sam output
```

Funkcja: validate_order_schema()

Input: Order JSON

Output: PASS / FAIL

Czas: 12ms

Czy skrypt działa poprawnie?

```
schema valid, required fields, types correct
```

WALIDACJA

- [V] validate_order_schema() wykonany
12ms, exit code 0
- [V] Schema poprawna
customer_id, items, total -- present
- [V] Typy danych prawidłowe
total: float, items: list
- [V] markpact:log zaktualizowany
STEP_COMPLETED: parse_order

5. Step: check_inventory

actor: llm -- AI sprawdza magazyn

README.MD -- MARKPACT:LOG +1

```
# Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
 name: order-api
 steps:
 - name: parse_order
 actor: script
 - name: check_inventory
 actor: llm
 - name: fraud_detection
 actor: script
 - name: human_payment_approval
 actor: human
 config: {channel: web, timeout: 7200}
 - name: process_payment
 actor: script
 - name: deploy_update
 actor: llm
```

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```

```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
 return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
 return {"status": "ok"}
```

```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

```yaml markpact:deploy
target: docker
pactown:
 name: order-api
 services:
 api: {port: 8000, health_check: /health}
```

```json markpact:state
{"phase": "running", "health": null, "success_count": 0, "error_count": 0}
```

```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

## OCZEKIWANIE: LLM POPRAWNIE GENERUJE

### [ACTOR: llm] AI (Ollama/OpenRouter)

Model: qwen2.5-coder:7b  
Input: Order items + inventory  
Output: Availability check  
Czas: 850ms

### Czy LLM poprawnie generuje?

valid JSON, items checked, no hallucinations

## WALIDACJA

- [V] LLM odpowiedzial  
850ms, model=qwen2.5-coder:7b
- [V] Response: valid JSON  
(all\_available: true, items\_checked: 3)
- [V] Wszystkie produkty sprawdzone  
3/3 items verified
- [V] Brak hallucinations  
Output matches inventory DB

# 6. Step: fraud\_detection

actor: script -- Algorytm fraud

## README.MD -- MARKPACT:LOG +1

```
Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
  name: order-api
  steps:
    - name: parse_order
      actor: script
    - name: check_inventory
      actor: llm
    - name: fraud_detection
      actor: script
    - name: human_payment_approval
      actor: human
      config: {channel: web, timeout: 7200}
    - name: process_payment
      actor: script
    - name: deploy_update
      actor: llm
  ...

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
    return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
    return {"status": "ok"}
```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```yaml markpact:deploy
target: docker
pactown:
  name: order-api
  services:
    api: {port: 8000, health_check: /health}
```json markpact:state
{"phase": "running", "health": null, "success_count": 0, "error_count": 0}
```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

OCZEKIWANIE: FRAUD CHECK

[ACTOR: script] Deterministyczny algorytm

Funkcja: run_fraud_check()
Sprawdza: IP, historia klienta, kwota
Risk score: 0.12 (low)
Czas: 45ms

WALIDACJA

- [V] run_fraud_check() wykonany
45ms, risk_score=0.12
- [V] Risk score < threshold (0.7)
0.12 < 0.70 -> SAFE
- [V] Klient nie na blacklist
customer_id not in blacklist

7. Step: human_payment_approval

actor: human -- PIPELINE ZABLOKOWANY

README.MD -- MARKPACT:STATE = BLOCKED

Order Management API

REST API z human approval przed platnoscia.

```
```yaml markpact:orchestration
```

```
pipeline:
 name: order-api
 steps:
 - name: parse_order
 actor: script
 - name: check_inventory
 actor: llm
 - name: fraud_detection
 actor: script
 - name: human_payment_approval
 actor: human
 config: {channel: web, timeout: 7200}
 - name: process_payment
 actor: script
 - name: deploy_update
 actor: llm
 ...
```

```
```json markpact:deps
```

```
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
  ...
```

```
```python markpact:file=app/main.py
```

```
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
 return {"id": "ORD-001", "status": "pending"}
```

```
@app.get("/health")
def health():
 return {"status": "ok"}
 ...
```

```
```bash markpact:run
```

```
uvicorn app.main:app --host 0.0.0.0 --port 8000
  ...
```

```
```yaml markpact:deploy
```

```
target: docker
pactown:
 name: order-api
 services:
 api: {port: 8000, health_check: /health}
 ...
```

```
```json markpact:state
```

```
{"phase": "blocked", "health": "waiting_human", "success_count": 0, "error_count": 0}
  ...
```

```
```text markpact:log
```

```
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

## OCZEKIWANIE: DECYZJA CZLOWIEKA

**[ACTOR: human] -- BLOCKED**

**Pipeline CZEKA na ludzka decyzje.**

Zamowienie \$2,500 > limit \$1,000.

```
HumanTask: task-abc123
Kanal: web (dashboard)
Timeout: 7200s (2h)
Powiadomienie: webhook -> Slack
```

Kanaly komunikacji:

[Dashboard] [Email] [Webhook] [Shell]

POST /api/pipeline/tasks/task-abc123

```
{action: "approve", by: "manager@co.com"}
```

## WALIDACJA

- [V]** HumanTask utworzony  
task-abc123, channel=web
- [V]** Webhook wyslany  
POST slack -> 200 OK
- [V]** markpact:state = blocked  
phase: running -> blocked
- [?]** Oczekiwanie na human...  
timeout za 7155s

# 8. Human -> Approve

Manager zatwierdza -- pipeline kontynuuje

## README.MD -- MARKPACT:STATE = RUNNING

```
Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
  name: order-api
  steps:
    - name: parse_order
      actor: script
    - name: check_inventory
      actor: llm
    - name: fraud_detection
      actor: script
    - name: human_payment_approval
      actor: human
      config: {channel: web, timeout: 7200}
    - name: process_payment
      actor: script
    - name: deploy_update
      actor: llm
  ...

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
    return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
    return {"status": "ok"}
```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```yaml markpact:deploy
target: docker
pactown:
  name: order-api
  services:
    api: {port: 8000, health_check: /health}
```json markpact:state
{"phase": "running", "health": null, "success_count": 0, "error_count": 0}
```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

CZLOWIEK PODJAL DECYZJE

APPROVAL RECEIVED

Decyzja: APPROVED
Przez: manager@company.com
Czas decyzji: 45 sekund
Kanal: Dashboard (web)

Zmiana w README.md:

[phase: blocked] --> [phase: running]

Pipeline kontynuuje automatycznie.

WALIDACJA

- [V] Human approval received
APPROVED by manager@company.com (45s)
- [V] HumanTask resolved
task-abc123 status=resolved
- [V] Pipeline wznowiony
status: blocked -> running
- [V] markpact:state zaktualizowany
phase: blocked -> running

9. Deploy -> Production

Ostatnie kroki + deployment

README.MD -- MARKPACT:STATE = DEPLOYED

Order Management API

REST API z human approval przed platnoscia.

```
```yaml markpact:orchestration
```

```
pipeline:
 name: order-api
 steps:
 - name: parse_order
 actor: script
 - name: check_inventory
 actor: llm
 - name: fraud_detection
 actor: script
 - name: human_payment_approval
 actor: human
 config: {channel: web, timeout: 7200}
 - name: process_payment
 actor: script
 - name: deploy_update
 actor: llm
 ...
```

```
```json markpact:deps
```

```
{ "runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"] }
```

```
```python markpact:file=app/main.py
```

```
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
 return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
 return {"status": "ok"}
...
```
```

```
```bash markpact:run
```

```
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

```
```yaml markpact:deploy
```

```
target: docker  
pactown:  
  name: order-api  
  services:  
    api: {port: 8000, health_check: /health}  
...  
```
```

```
```json markpact:state
```

```
{ "phase": "deployed", "health": "ok", "success_count": 1, "error_count": 0 }
```

```
```text markpact:log
```

```
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```

## OCZEKIWANIE: DEPLOY POPRAWNY

Pozostale kroki:

[V] process\_payment -- PASS (320ms)

[V] deploy\_update -- PASS (1200ms)

**Pipeline COMPLETED -- 6/6 steps passed**

Finalny stan README.md:

[phase: running] --> [phase: deployed]

## WALIDACJA

[V] process\_payment -> PASS

320ms, transaction\_id=TXN-001

[V] deploy\_update -> PASS

1200ms, container started

[V] Healthcheck /health -> 200 OK

{status:ok}

[V] markpact:state.phase = deployed

success\_count: 0 -> 1

[V] markpact:log -- 10 wpisow

Pełna historia pipeline

# 10. Self-Learning

Pattern zapisany -> kolejny projekt lepszy

## README.MD -- FINALNY STAN

```
Order Management API

REST API z human approval przed platnoscia.

```yaml markpact:orchestration
pipeline:
  name: order-api
  steps:
    - name: parse_order
      actor: script
    - name: check_inventory
      actor: llm
    - name: fraud_detection
      actor: script
    - name: human_payment_approval
      actor: human
      config: {channel: web, timeout: 7200}
    - name: process_payment
      actor: script
    - name: deploy_update
      actor: llm
```

```json markpact:deps
{"runtime": "python:3.12", "packages": ["fastapi", "uvicorn", "pydantic"]}
```

```python markpact:file=app/main.py
from fastapi import FastAPI
app = FastAPI(title="Order API")

@app.post("/orders")
def create_order(body: dict):
    return {"id": "ORD-001", "status": "pending"}

@app.get("/health")
def health():
    return {"status": "ok"}
```

```bash markpact:run
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

```yaml markpact:deploy
target: docker
pactown:
  name: order-api
  services:
    api: {port: 8000, health_check: /health}
```

```json markpact:state
{"phase": "deployed", "health": "ok", "success_count": 1, "error_count": 0}
```

```text markpact:log
[T+0s] CONTRACT_CREATED
[T+1s] PIPELINE_STARTED
... (kontynuacja)
```
```

## EWOLUCJA NA BAZIE DANYCH

Pattern Library -- zapisany wzorzec:

```
id: api-rest-orders
keywords: rest, api, orders, payment
success_rate: 1.00
usage_count: 1
service_type: api
```

Następny podobny prompt:

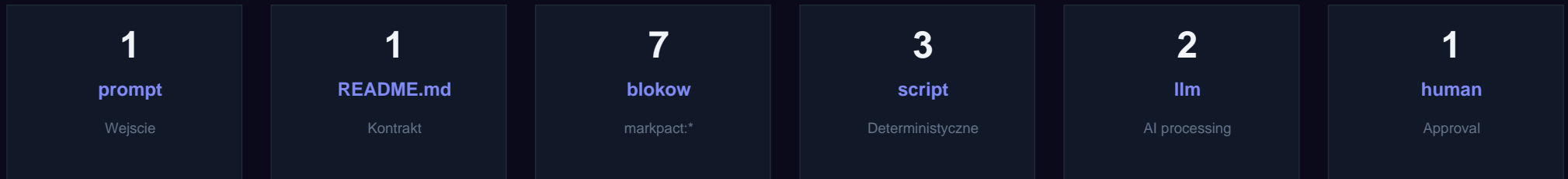
System znajdzie ten pattern i użyje go jako template. Success rate rośnie z każdym sukcesem.

## WALIDACJA

- [V] `PatternLibrary.save_from_contract()`  
Pattern api-rest-orders zapisany
- [V] `success_rate = 1.00`  
Pierwszy run -> 100%
- [V] `Pattern reusable`  
Kolejny 'REST API + orders' -> match

# Podsumowanie

Cały przepływ w jednym pliku README.md



**Jeden plik. Jeden kontrakt. Wszystko walidowalne.**