Use Case #2: Augment Product Owners

💡 Idea in brief: Product Owner Copilot

1.1. What problem are we solving?

> How might we reduce the time and effort Product Owners spend on coordination, reporting, and re-planning across multiple teams, while improving predictability and visibility?

Each ODC product initiative involves 10–15 asset-centric teams. The 2nd order effects of this are pretty impactful: - --

- Heavy coordination overhead.
- POs today manage coordination, dependency mapping, reporting, and plan re-adjustments manually.
- Planning & dependency tracking is manual, with repeated sync meetings.
- Plans shift often (“life happens!”), requiring constant re-alignment and risk mitigation.
- This slows delivery, adds friction, and keeps POs focused on admin instead of strategy.

1.2. Why should we solve now?

- Multi-team dependency complexity is rising as ODC expands.
- POs spend disproportionate time on reporting vs. value delivery.
- AI copilots are now mature enough to automate translation, reporting, and risk detection.
- A staged approach delivers quick wins and builds towards a strategic PO AI Copilot.

Here's how "Product Owner Copilot" fits into our prioritisation framework:

Criteria	Score
Impact on metric	Medium
Time to deliver	Medium (~4–6 weeks for staged PoC)
Feasibility	Medium-High
Adoption likelihood	High (reduces low-value admin burden)

1.3. Assumptions

- AI copilots can learn to translate business <> technical requirements with fine-tuning given they have access to the right context (supermodular.ai team has already proven this outside OutSystems).
- Jira/Epics and reporting data are available for AI ingestion.
- Additional context for models can be quickly tested via vector database ingestion, not deep integration.
- POs will adopt if copilots save significant time and reduce friction.

- PO metrics (predictability, throughput variance, reporting time) can be tracked and there's an existing baseline.

🏗️ 2. Implementation Suggestion

2.1. What it is | PoC Tech Approach

We propose testing a staged value delivery:

1. **Business Requirements <> Technical Requirements Translation**
    - **Assumption to prove:** POs don't need full context of the technical constraints and environment to move the execution process along
    - **What:** Auto-convert business/product requirements into dev-ready technical specs.
    - **How: TO ADD**
    - **Metrics Impact:**
        - Sprint Predictability (≥ 80%)
        - Throughput Variance (≤ 20%)
2. **Auto-generated Progress Reports**
    - **Assumption to prove:** POs don't need to spend time crunching data and merging data sources to provide visibility on key metrics and progress
    - **What:** Generate initiative/epic status reports from Jira to cut down reporting time.
    - **Metrics Impact:**
        - PO time on support tasks (baseline vs. post-AI)
        - Value Stream Metrics achieved (≥ 75%)
3. **Risk & Dependency Alerts**
    - **Assumption to prove:** AI can augment humans on being aware of project constraints, dependencies and technical edge cases ahead of implementation
        - This can be done once #1 and #2 are proved possible.
    - **What:** Flag conflicts, delays, or risks across multi-team dependencies.
    - **Metric Impact:** Delivery deviation in GA Date (≤ 15%)

2.2. What it is not

- This isn't a replacement for the PO role or strategic decision-making.
- This isn't a one-off reporting bot. It's designed to evolve into a full copilot.

3. Impact and Criteria for Success

- **Primary leading metric:** PO time spent on coordination & reporting.
- **Secondary metrics:**
    - Sprint Predictability (≥ 80%)
    - Throughput Variance (≤ 20%)
    - Delivery deviation GA Date (≤ 15%)
    - Friction Score (< 60)
- **Lagging metric:** Improved delivery flow & reduced missed dependencies.

4. Risks and Known Issues

- AI accuracy in translating requirements requires iteration. We'll see value compounding over time, not overnight.
- Dependencies data might be siloed across tools or live in people's brains, not systems, capping value-added from AI
- Building a PoC in Jira (for ex for the Business<>Requirements milestone) is feasible given we have permissions. But to deliver true value it's likely the model will need additional context which requires integration/custom work that might not be feasible in just a few weeks.