# Reasoning on Synthetic Data, AI Engineering, and Agentic Licensing

Prof. Dr. Petre Dini, ex-AT&T/Cisco Systems, Inc., USA (retired)
Prof. Dr. Lasse Berntzen, University of South-Eastern Norway, Norway

# Themes



- (early) Marketing Decisions
- Business Cases based on Synthetic Data
- Challenges of C-Series Coordination
- Early Delivery vs On Time vs Perfection
- Prosecutive views
- Human Creativity
- Revenue Streams and Licensing
- Virtual AI Engineering and Agentic Frameworks

# Marketing Decisions

PORTO 2026

- Marketing decisions
- Paradigm change on nature of data

- Utility (need): Proof of Market
- Feasibility: Proof of Concept
- Usability: Proof of Customer Adoption

- Use Cases Proven Catalog
- Scale of users: niche/large

- Perenniality (noun): The quality or condition of being perennial.
- Perennity (noun): Lastingness; perennial duration; the state of being perennial.

- Technology-agnostic needs
- Technology-related needs

- Based on existent/persistent needs
- Based on potential/foreseen/predicted needs

- Individual-based user case, vs. critical mass-user case

**Business cases should be derived from a need that lasts supported by a critical mass of users for a lasting duration**

# Use of Synthetic Data

PORTO 2026

We are not only generating synthetic data, but also, we are generating **synthetic problems** and **synthetic value**.

**If the use case is artificial, the success is artificial (synthetic value ≠ from real/market value**

**What and how is happening?**

AI enables **capability-first thinking**

Demos replace **real constraints**

Validation happens in **controlled environments**

**What looks like success: clean workflows / impressive output / fast prototyping**
**What is missing: real user pain / operational constraints / measurable impact / accountability**

**Key Risks**

Strategic drift → roadmap follows demos
False validation → "it works" ≠ "it matters"
Hidden costs → integration, data, verification
Crowded priorities → real problems ignored

**Quick Diagnostic**

Who needs it?
Who pays for it?
What decision improves?
What happens when it fails?

**AI makes it easy to fabricate usefulness.**
**Real value still requires real problems.**

**Reality vs Fabrication**

# Downsides of Using Synthetic Data

PORTO 2026

## Q1. Reality vs Fabrication
How can we distinguish between a genuine use case and one engineered to showcase AI capability?

## Q2. Demo Trap
Are organizations optimizing for what can be demonstrated rather than what should be solved?

## Q3. Validation Illusion
If a use case works in a controlled environment, what guarantees its relevance in real operations?

## Q4. Ownership and Accountability
Can a use case be considered real if no stakeholder clearly owns its outcome or failure?

## Q5. Strategic Risk
Do synthetic use cases delay transformation by creating the illusion of progress?

## Q6. Investment Discipline
Should organizations require “problem-first justification” before approving AI initiatives?

**The most dangerous side of the success of a synthetic use case is the one solving a problem nobody had before.**

# Challenges of C-series Coordination

PORTO 2026

- **Design vs validation vs selling**
  - Different objectives (user, codesign, feedback)
- **Partial vs Full feature set**
  - Pioneering or Loosing competition
- **Target domain vs all domains**
  - Use where it works, postpone where it doesn't
- **Competition, Pioneering, Market scale**
  - **ROI – vs U {roi}**
    - Compensatory areas & Success
  - **Corporate ROI vs Society ROI**
    - Paradigm change on society metrics
- **2025 MIT NANDA Report should be a wake-up call for every C-suite**
  - **5% AI success rate**
- **Engineering vs Management**
  - 95% Eng & 5% C-suite
- **Operational & Maintenance**
  - Selling too early and unproven benefits are detrimental
  - Selling Just in Time
  - Consider continuum in development

# Q&A

**How to assess the risk for a (too) early delivery?**

[technology - digital, social media - platforms, products – smartphones]

**How to control the exposure of hallucinations when using AI-tools?**

[use it where it works, knife paradigm]

**How to balance feedback/creativity/versioning vs too-late-but-perfect delivery?**

**What metrics should be used at the Society C-suite level?**

**How to mitigate the risk of adopting an early phase of a fledgling technology?**

# Prospective views



- **In an overview**
  **(https://www.iaria.org/conferences2025/filesNetWare25/Tutorial_PetreDini_TheAnatomyOfAgenticFramew.pdf ),**
- **I identified that there are a lot of things that we know and that work. It takes time for corporations to see the major benefit of a technology (including AI-based ones); encouragingly, I've seen academic curricula adapted, new dedicated business units in medium/big corporations, new engineering area (prompting, licensing, etc.), name it, Engineering. Incredibly, these are totally different skills than a traditional engineering. This triggers the call for avoiding a quick evaluation in the same time of preparing skilled workforce.**
- **My interpretation of the report in discussion is that only 5% of management teams understood where and how to use any AI-based tools. This seems to me being unfair to engineering achievements.**
- **To clarify, management teams have no choice; they are pushed by the Boards, Boards are pushed by Investors, and ..... ; if we want to generalize any technology (including all AI-based tools) at the society level, we should have appropriate metrics at the society level. Those 95% had metrics at the corporate level, based on unrealistic estimations driven by business-plans. Therefore, those estimations should be tuned.**

## Just in Time Often Beats Perfection

- **Action Over Inaction:** Perfectionism often leads to paralysis or excessive delays, whereas JIT, or "good and on time," focuses on delivering value when it is needed.
- **Reduced Costs and Waste:** JIT reduces the need for excess inventory and "overproduction" of quality, saving time, money, and resources that would otherwise be wasted in trying to make something flawless.
- **Agility and Flexibility:** JIT allows for faster responses to changing market demands or customer needs, which is crucial in fast-paced environments.
- **Continuous Improvement (Kaizen):** JIT fosters a culture of, "never let perfect get in the way of done," which encourages consistent, incremental improvements rather than waiting for a perfect, final product.

## Why Perfection is Difficult (The Role of Human Creativity)

- **Unattainable Standard:** Perfection is generally an illusion, as it is subjective and impossible to achieve, particularly when dealing with complex, creative, or constantly evolving tasks.
- **Creativity and Spontaneity:** Human creativity is dynamic and messy, often disrupted by the rigid, self-critical, and judgmental nature of perfectionism.
- **Diminishing Returns:** The effort required to turn a "good" product into a "perfect" one often costs far more in time and resources than the marginal value added.

## Key Takeaways

- **"Done is better than perfect":** Focusing on finishing tasks allows for feedback and iteration, while waiting for perfection often results in never finishing at all.
- **"Good and on time":** This is often the superior standard to "perfect and late," as the latter can lead to frustration and missed opportunities.
- **Embrace Imperfection:** Accepting that mistakes are part of the process fosters a more productive and, ironically, more creative environment

## Human Creativity is fluid "Just in Time" (JIT) usually wins out over "Perfection":

### 1. The Cost of the "Final 10%"

In many projects, reaching 90% completion takes 50% of the effort. Reaching that final 10% to achieve "perfection" often takes the remaining 50% of the time.

**JIT:** Delivers value while the need is still relevant.

**Perfection:** Risks delivering a "flawless" solution to a problem that no longer exists.

### 2. The "Moving Target" of Creativity

Human creativity means we are constantly iterating. If you wait for perfection, you are trying to freeze a snapshot of a moving train.

**Adaptability:** JIT allows for "course correction." You release a version, see how it performs, and use your next creative spark to improve it based on real-world feedback.

**Stagnation:** Perfectionism often leads to "analysis paralysis," where fear of a flaw prevents any progress at all.

### 3. Feedback Loops vs. Assumptions

Perfection is often based on **internal assumptions** (what *we* think is best). JIT is based on **external reality**.

**Real-world testing:** Shipping "just in time" provides data. You learn what actually works rather than what you *imagined* would work.

**Resource Management:** It prevents wasting creative energy on features or details that the end-user might not even notice or care about.

# C-series variety (i)

PORTO 2026

## Core / Traditional (found in most organizations)

- **CEO** — Chief Executive Officer
- **COO** — Chief Operating Officer
- **CFO** — Chief Financial Officer
- **CMO** — Chief Marketing Officer
- **CIO** — Chief Information Officer
- **CTO** — Chief Technology Officer

## Technology / Data / Security / Digital

CISO — Chief Information Security Officer
CDO — Chief Data Officer
CDAO — Chief Data & Analytics Officer
CDO (Digital) — Chief Digital Officer
CAIO / CDAO (AI) — Chief AI Officer / AI & Data variants
CTO — (also product technology leadership)
CXO — Chief Experience Officer (customer experience)

## Very Common Functional Chiefs

CHRO / CPO (People) — Chief Human Resources / People Officer
CLO — Chief Legal Officer (sometimes Chief Learning Officer — context-dependent)
CSO — Chief Strategy Officer
CRO — Chief Revenue Officer
CCO — Chief Commercial Officer or Chief Communications Officer
CAO — Chief Administrative Officer
CBO — Chief Business Officer
CPO — Chief Product Officer / Procurement Officer

## Commercial / Customer / Market Roles

CCO (Customer) — Chief Customer Officer
CRO — Chief Revenue Officer
CSO — Chief Sales Officer
CBO — Chief Brand Officer
CBDO — Chief Business Development Officer

# C-series Variety (ii)

PORTO 2026

## Governance / Risk / Compliance

- **CCO (Compliance)** — Chief Compliance Officer
- **CRCO** — Chief Risk & Compliance Officer
- **CRO (Risk)** — Chief Risk Officer
- **CIGO** — Chief Information Governance Officer

## Strategy, Innovation, Transformation

- **CINO** — Chief Innovation Officer
- **CTrO** — Chief Transformation Officer
- **CVO** — Chief Visionary / Value Officer
- **CIDO** — Chief Idea Officer (innovation-focused role)

## ESG / Sustainability / Culture / People

- **CSO (Sustainability)** — Chief Sustainability Officer
- **CGO** — Chief **CHO** — Chief Happiness OfficeGreen Officer
- **CDO (Diversity)** — Chief Diversity Officer
- **CWO** — Chief Well-Being Officer

## Operations, Supply, Facilities

- **CSCO** — Chief Supply Chain Officer
- **CPO (Procurement)** — Chief Procurement Officer
- **CLO (Logistics)** — occasionally used
- **CFOO / COO variants** — operations specialties

## New / Trend-Driven / Experimental Titles

(Especially in tech companies and startups)

- Chief AI Officer
- Chief Platform Officer
- Chief Ecosystem Officer
- Chief Metaverse Officer
- Chief Trust Officer
- Chief Ethics Officer
- Chief Remote Officer
- Chief Community Officer

## Specialized / Sector-Specific Roles (Increasingly Seen)

- **CAO (Accessibility)** — Chief Accessibility Officer
- **CMO (Medical)** — Chief Medical Officer (healthcare)
- **CNO** — Chief Nursing Officer
- **CISO (Physical Security)** — sometimes Chief Security Officer
- **CPO (Privacy)** — Chief Privacy Officer
- **CAgO** — Chief Agriculture Officer (agribusiness)
- **CAvO** — Chief Aviation Officer

# Questions on the Table

## Q&A

**How to assess the risk for early delivery?**

**How to control the exposure of hallucinations when using AI-tools?**

**How to balance feedback/creativity/versioning vs useless perfect delivery?**

**What metrics should be used at the Society C-suite level?**

**How to mitigate the risk of adopting an early phase of a fledgling technology?**

# Revenue Streams and Licensing

PORTO 2026

## Revenue streams

| | |
|---|---|
| 💼 B2B Licensing | License the system as a SaaS or on-prem platform (monthly/annual tiers) |
| 🛠️ Customization Services | Custom workflows or modules tailored to client processes |
| 🔁 API as a Service | Expose SME-layer functionality via custom APIs |
| 🧩 Integration Add-ons | Plugins for CRMs, ERPs, sector-specific software (education, law, medical) |
| 📑 Analytics / Reporting | Premium dashboards or compliance modules |

CRM Customer Relation Management

ERP Enterprise Resource Planning

## Licensing strategy

Use tools under licenses that allow derivative/commercial work or that are “as-a-service” (API-based)

| *Tool Type* | *Example* | *SME Strategy* |
|---|---|---|
| 🧠 LLMs | OpenAI, Claude, Cohere | Use via API – no model redistribution |
| 🧰 Open-source libs | spaCy, HuggingFace, FastAPI | Comply with licenses (MIT/Apache = safe) |
| 📦 Open-core | LangChain, Supabase | Use core, pay for commercial support if needed |
| 💻 Cloud APIs | AWS Comprehend, Azure ML | Usage-based costs; no IP lock-in |

## Be careful with:

GPL code (may require open-sourcing your product)

Redistributable binaries if licensing terms are restrictive

GNU GPL: General Public License (GPL)

TP-LINK GPL code: Third Party : partly contain software code developed by third parties, including software code subject to the GNU General Public Licence (“GPL“), Version 1/Version 2/Version 3 or GNU Lesser General Public License("LGPL")

# Roles, Tools, Licensing

PORTO 2026

## Team Roles & Assignments

| *Role* | *Name / Notes* | *Time Allocation* |
|---|---|---|
| 🧠 AI Architect / Engineer | | Full-time / Part-time |
| ✍️ Prompt Designer / Evaluator | | |
| 🧩 Domain Expert / Product Owner | | |
| 💻 Full-stack Developer | | |
| ⚖️ Legal / Compliance Advisor (optional/external | | |

## Core Tools & Frameworks

| *Tool Type* | *Selected Tool* | *Why This Tool?* |
|---|---|---|
| LLM API | (e.g., OpenAI GPT-4, Claude, Mistral) | |
| Concept Engine / Graph | (e.g., Neo4j, RDF) | |
| Prompt Chains / Lang | (e.g., LangChain, CrewAI) | |
| Embedding Store | (e.g., Chroma, Pinecone, FAISS) | |
| UI / App Framework | (e.g., React, Streamlit, FastAPI) | |

## Iteration & Evaluation Plan

| *Evaluation Focus* | *Approach / Metric* | *Frequency* |
|---|---|---|
| Output Quality | Human eval + metrics (BLEU/Faithfulness) | Weekly |
| User Feedback | Real-user pilot, scorecards | Monthly |
| Cost Optimization | Token usage monitoring | Every sprint |

## Risk & Licensing Checklist

| *Area* | *Action Required* | *Status* |
|---|---|---|
| API Terms | Review commercial use terms | ☑ |
| Data Privacy | Compliant with GDPR / local regs | ☐ |
| Open Source | Track LGPL/GPL obligations | ☐ |
| Prompt/IP Boundaries | Avoid sensitive reuse | ☐ |

## Core Capabilities of a Virtual "AI Engineer"

| Capability | Description |
|---|---|
| 🧱 Model Architecting | Selects models based on data type (e.g., CNNs, LLMs, GNNs) |
| 🧪 Hyperparameter Tuning | Uses optimization techniques (Bayesian, Grid Search) |
| ⚙️ Pipeline Automation | Builds end-to-end data/model training workflows |
| 📦 Deployment Automation | Pushes models into staging or production environments |
| 📝 Documentation & Reporting | Auto-generates experiment reports, code comments |
| 🔁 Continual Learning Mgmt | Suggests or implements retraining schedules |
| 🔍 Debugging Assistant | Identifies performance regressions, training bugs |
| 🔐 Bias and Compliance Checks | Flags fairness, privacy, or explainability issues |

### Underlying Technologies

> 🧠 LLMs | GPT-4, Claude, Mistral, or open-source models (fine-tuned for engineering tasks)

> 🕸️ Agent Frameworks | LangChain, AutoGen, CrewAI, AgentVerse

> 🛜 Tool Plugins | Code runners, databases, version control, Docker, etc.

> ⚙️ Memory/Planning | ReAct, Chain-of-Thought, RAG, scratchpads, vector memory

> 📁 Environment | Often containerized (Docker, Replit, VSCode in-browser)

## Examples of Platforms Creating "AI Engineer" Entities

> Devin by Cognition

A fully autonomous AI software engineer that can plan, write, debug, and test code with no human intervention (still in preview

> GitHub Copilot X (with Agents)

Goes beyond code completion; can scaffold apps, generate entire classes, and collaborate over time.

> AutoGPT / AgentGPT

Experimental open-source agents that can be instructed to achieve high-level engineering goals via tool use and iterative planning.

> OpenDevin (open-source fork)

Tries to mimic Devin's architecture — acts like a terminal-based AI engineer that uses planning + code execution.

> C> odeWhisperer (AWS) and Tabnine

Autocomplete-style assistants but heading toward semi-autonomous behavior.

> LangChain Agents / CrewAI

Build modular agent teams: one can play the "AI engineer" role in an LLM-powered workflow.

# Agentic Engineering

PORTO 2026

An "**AI Engineer**" as a virtual agent (an autonomous LLM-based entity that performs AI engineering tasks),
**Agentic Engineering** as a discipline or paradigm (engineering systems of agents that plan, act, and learn over time).

## AI Engineer (as a virtual agent)

This is a specialized role or embodied skillset within a broader system.
It refers to an LLM-powered autonomous agent that:
- Writes code, designs models, debugs, deploys
- Acts like a virtual software engineer
- Is task-focused (e.g., “build me an object detector”)
- May use planning, tool use, memory, and execution environments (e.g., shell, browser, Python interpreter)

Example:

Devin, GitHub Copilot + agents, or a LangChain/CrewAI agent with the "AI Engineer" role.

## Agentic Engineering

This is a new field of engineering focused on the design, orchestration, and safety of intelligent agents, especially LLM-based ones.
It involves:
- Creating multi-agent ecosystems
- Managing goals, delegation, planning, negotiation
- Enforcing safety, alignment, and controllability
- Addressing non-determinism, long-horizon actions, and memory evolution

## Related challenges include:

- Tool integration
- Agent teaming and coordination
- Goal disambiguation and intent refinement
- Autonomy vs. oversight balancing
- Temporal abstraction (short tasks vs. lifelong learning)

# AI vs Agentic Engineering



Their Relationship

| Aspect | AI Engineer (Agent) | Agentic Engineering |
|---|---|---|
| 🎭 What | A software agent that performs AI tasks | A discipline to build, manage, and evaluate agents |
| 🧠 Cognitive Role | Acts as a specialized skill worker | Designs systems of cognition and delegation |
| 🧰 Technologies Used | LLMs, memory, RAG, tool APIs | Agent platforms, safety modules, coordination logic |
| 🛠️ Output | Trained models, deployed pipelines | Robust multi-agent systems, reliable interfaces |
| 🔄 Scope | One agent with a fixed or growing role | Multi-agent environments, emergent behaviors |
| 🧩 Example | Devin generating a neural net | CrewAI orchestrating 5 agents for research |

**What Is Goal Generation in Agentic Systems?**

**Goal generation is the process by which an agent (like the AI Engineer) determines what it should do next. This includes:**

- Recognizing new needs or opportunities
- Transforming open-ended tasks into actionable objectives
- Aligning tasks with long-term system purpose or constraints

**1. Components of Goal Generation**

| Source | Example |
|---|---|
| 👩 Human Prompt | "Build a model to classify pneumonia in X-ray images" |
| 🧠 Self-reflection | Agent detects pipeline drift and sets a goal to retrain |
| 📈 Environment State | New data availability triggers model update |
| 🔄 Upstream Agent | A supervisor agent delegates "optimize hyperparameters" |

# Qs: Agentic Framework



How to align business goals, problem solution, and agents' goals?

Automatic Goal Generation?

Similarity with what an LLM produces these days, e.g., the plans they produce?

How would you formalize this?

Ho to ensure they solve the problem?

What about outcome quality?

What about conflicting goals, the design trade-offs?

Establishing robust infrastructure and processes