Technical Blueprint: Autonomous Technology Forecasting and Patent Analytics Platform

1. Executive Vision and System Architecture

The global landscape of Research and Development (R&D) is currently undergoing a fundamental transformation, shifting from intuition-based strategy to evidence-based forecasting. As technological complexity accelerates, the ability to predict dominant designs and identify obsolescence before it occurs has become the primary differentiator for competitive advantage. This report serves as a comprehensive technical blueprint for constructing a sovereign, sovereign-grade technology forecasting platform designed to rival and surpass incumbent solutions such as GetFocus/Odin. The proposed system leverages the rigorous scientific methodologies of Benson, Magee, and Trancik to quantify the rate of technological improvement (TIR), integrated within a scalable, agentic AI architecture capable of processing global patent corpuses in near real-time.[1]

1.1 The Strategic Imperative for Quantitative Forecasting

Traditional approaches to patent analytics—primarily based on volume counting and keyword co-occurrence—have proven insufficient for predicting technological outcomes. High patent volume often correlates with defensive legal strategies rather than genuine innovation. To provide "boardroom-ready" insights, as promised by competitors like GetFocus, a platform must measure the *quality* and *velocity* of innovation, not just its quantity.[1]

The "Blueprint" detailed herein is founded on the premise that technological evolution follows discernable, mathematical laws. By analyzing the directional flow of knowledge (via citations) and the structural complexity of designs (via classification overlap), we can calculate a Technology Improvement Rate (TIR) that serves as a proxy for performance/cost ratios. This allows the system to generate S-curve forecasts that predict when a specific technology (e.g., Solid State Batteries) will cross the inflection point and displace an incumbent (e.g., Li-ion).[2]

1.2 Architectural Philosophy: The Agentic RAG Paradigm

To achieve the necessary scale and reasoning capability, the System rejects the traditional static database model in favor of an **Agentic Retrieval-Augmented Generation (Agentic RAG)** architecture. In this paradigm, the "user" does not simply query a database; they interact with a fleet of autonomous software agents. These agents are empowered to decompose high-level strategic questions (e.g., "Find the whitespace in hydrogen storage") into executable sub-tasks, traversing massive vector indices and knowledge graphs to synthesize answers.[5]

This architecture addresses the "Context Window" limitation of Large Language Models (LLMs). While modern LLMs have large contexts, stuffing 100,000 patents into a prompt is cost-prohibitive and technically flawed. Instead, our Agentic RAG system uses a **Knowledge Graph** to maintain the global context of the patent citation network, while using **Vector Search** for local semantic retrieval. This "GraphRAG" approach enables the system to answer "global" queries about technology trends that require traversing thousands of document relationships, a capability where standard RAG fails.[7]

Architectural Component	Traditional Analytics	Proposed System (The Blueprint)
Core Methodology	Patent Counting / Keywords	Benson-Magee TIR / Cycle Time / Trancik Complexity
Data Retrieval	Boolean Search / SQL	Semantic Vector Search + Graph Traversal (GraphRAG)
Reasoning Engine	Human Analyst	Autonomous Agentic Workflow (LangGraph/LangChain)
Forecasting Model	Linear Extrapolation	Fisher-Pry Transform (S-Curve) & Logistic Regression
Infrastructure	Monolithic Database	Distributed Microservices (Milvus/Neo4j/K8s)

1.3 System High-Level Design

The platform is composed of four distinct, loosely coupled layers, each designed for independent scaling:

1. **The Ingestion & Normalization Layer:** A distributed fleet of Python-based extraction agents responsible for harvesting raw bulk data from the USPTO (United States), EPO (Europe), and WIPO (World) repositories. This layer handles the complex parsing of legacy formats (Greenbook/Redbook), optical character recognition (OCR) corrections, and API throttling management.[9]
2. **The Knowledge Processing Layer (KPL):** The computational heart of the system. Here, raw text is transformed into high-dimensional vector embeddings for semantic search (using models like text-embedding-3-small or domain-specific BERT) and structured nodes/edges for the Knowledge Graph. This layer also computes the heavy bibliometric indicators: Search Path Link Count (SPLC) and Main Path Analysis.[11]
3. **The Forecasting Engine:** A statistical modeling module that fits the historical data generated by the KPL to logistic growth curves. It applies the Fisher-Pry transform to predict future saturation levels and cycle time acceleration, effectively generating the "TIR" metric.[13]
4. **The Insight & Experience Layer:** The user-facing interface that moves beyond static charts to interactive discovery. It features a "Chat with Patents" interface powered by the Agentic RAG, dynamic S-curve dashboards, and 2D projections of the

high-dimensional "whitespace" in the IP landscape.[3]

The following sections detail the engineering and scientific specifications for each of these layers, providing a roadmap from MVP (Minimum Viable Product) to a global-scale enterprise solution.

---

2. Scientific Core: The Mathematics of Innovation

The predictive power of the System relies entirely on the validity of its underlying mathematical models. We do not invent new metrics; rather, we operationalize the peer-reviewed methodologies of the MIT Technological Innovation/Improvement rates lab (specifically Benson, Magee, and Trancik). This scientific grounding provides the "boardroom-ready" confidence required by users.[16]

2.1 Quantitative Determination of Technological Improvement

The central metric of the platform is the **Technology Improvement Rate (TIR)**. As established by Benson and Magee (2015), the rate at which a technology improves (in terms of performance metrics like cost/watt or calculations/second) is strongly correlated with the citation statistics of the patents covering that technology.[18]

The System calculates the improvement rate (k) for a domain using a regression model that correlates patent metrics with historical performance data. The core equation derived from this research connects the *importance* of patents (citation counts) and their *immediacy* (recency).

2.1.1 The Fundamental Equation

For a given technology domain D, defined by a set of patents P_D, the estimated improvement rate k is calculated as:

$$k \approx \alpha \left(\frac{1}{|P_D|} \sum_{i=1}^{|P_D|} \text{Score}(p_i) \right) + \beta$$

Where $|P_D|$ is the Simple Patent Count (SPC). The $\text{Score}(p_i)$ is a composite metric. While simple forward citations (FC) are a primary component, the raw count is insufficient due to the "truncation problem"—recent patents haven't had time to accumulate citations. Therefore, the System utilizes a time-normalized citation score.[18]

The precise formulation implemented in the Forecasting Engine is:

$$TIR_{domain} = \frac{\sum_{t=T_{start}}^{T_{end}} \left(\text{MeanImmediacy}_t \times \text{MeanImportance}_t \right)}{T_{end} - T_{start}}$$

Importance is measured by the Forward Citation count relative to the average for that year and technology class (citation percentile). **Immediacy** is measured by the backward citation age (Cycle Time). The convergence of these two factors—highly cited patents that cite very recent prior art—is the strongest signal of rapid technological improvement.[18]

2.2 Cycle Time Analysis: Measuring the "Pulse" of Innovation

Cycle Time (CT) is the heartbeat of the forecasting model. It measures the temporal distance between a patent and the prior art it builds upon. Fast-moving technologies (e.g., AI, Semiconductors) exhibit short cycle times (3-5 years), while mature technologies (e.g., Civil Engineering) exhibit long cycle times (15-20 years).[3]
The System calculates the cycle time for every patent p as the median age of its backward citations (C_{back}):

$$CT_p = \text{Median}(\{ \text{Date}_{grant}(p) - \text{Date}_{grant}(c) \mid \forall c \in C_{back}(p) \})$$
However, simply averaging all backward citations introduces noise, as examiners often cite old "foundational" patents that are not indicative of the current pace. To correct for this, the System applies an **Exponential Decay Weighting** to the calculation, prioritizing recent citations which represent the "active front" of research.[23]
Weighted Mean Age (WMA):

$$WMA_p = \frac{\sum_{c \in C_{back}} (\Delta t_c \cdot e^{-\lambda \Delta t_c})}{\sum_{c \in C_{back}} e^{-\lambda \Delta t_c}}$$
Where Δt_c is the age of the cited patent c, and λ is a decay constant empirically derived from the specific IPC class (typically 0.1 to 0.3). A decreasing trend in the WMA for a technology domain signals an acceleration in innovation—a key predictive signal for the S-curve inflection point.[25]

2.3 Main Path Analysis and SPLC

To visualize the "River of Innovation"—the critical trajectory of a technology—the System employs **Main Path Analysis (MPA)**. In a citation network of 10 million nodes, finding the sequence of key inventions is non-trivial. The System uses the **Search Path Link Count (SPLC)** algorithm, widely regarded as the gold standard for identifying the "spine" of a citation network.[26]
The SPLC Algorithm Definition:
For every citation link $(u \to v)$ in the network:

1. Calculate $N_{sources}(u)$: The number of paths from all start nodes (patents with no backward citations in the set) to patent u.
2. Calculate $N_{sinks}(v)$: The number of paths from patent v to all end nodes (patents with no forward citations in the set).
3. The traversal weight of the link is: $SPLC_{u,v} = N_{sources}(u) \times N_{sinks}(v)$.

The "Main Path" is constructed by starting at the source node with the highest traversal count and following the link with the highest SPLC value at each step until a sink is reached. This mathematically identifies the sequence of patents that have facilitated the greatest flow of knowledge through the domain.[12]

2.4 Trancik's Design Complexity Theory

To explain *why* some technologies improve faster than others, the System integrates Jessika Trancik's theory of **Design Complexity**. This theory posits that the rate of improvement is constrained by the inter-connectivity of the system's components.[17]
The System estimates Design Complexity (D_C) by analyzing the **Classification Overlap**. A patent that cites references from a highly diverse set of IPC subclasses suggests a complex, interconnected design that may face "bottlenecks" in improvement. Conversely, modular designs (low overlap) tend to improve faster.[30]

$$Rate \propto \frac{1}{D_C}$$
By calculating the entropy of the IPC codes in the backward citation network, the System assigns a "Complexity Score" to each technology domain. This score serves as a damping factor in the S-curve forecast—highly complex technologies are predicted to have shallower adoption curves and slower cost declines.[31]

2.5 The Fisher-Pry Transform (S-Curve)

The culmination of these metrics is the **S-Curve Forecast**. Technologies follow a logistic growth pattern: incubation, exponential growth, and saturation. To predict where a technology is on this curve, the System fits the historical data (patent counts weighted by importance) to the Fisher-Pry model.[13]
The Fisher-Pry transform linearizes the S-curve, making it amenable to regression analysis:

$$\ln \left(\frac{f}{1-f} \right) = \alpha + \beta(t - t_0)$$
Where:

- f: The cumulative adoption fraction (estimated by cumulative weighted patents / estimated theoretical limit).
- t: Time (Year).
- β: The diffusion rate (slope), derived from the Cycle Time and Improvement Rate (k).
- t_0: The inflection point (time at 50% penetration).

By solving for β and t_0, the System generates the predictive dashboard, showing users exactly when a challenger technology is expected to overtake an incumbent.[32]

---

3. Data Engineering: The Global Ingestion Pipeline

To feed the scientific models, the System requires a comprehensive, continuously updated dataset of global innovation. This requires ingesting petabytes of unstructured text and metadata from the "Big 3" patent offices: USPTO (USA), EPO (Europe), and WIPO (World/PCT). This layer is the most engineering-intensive, requiring robust handling of legacy formats, throttling, and data normalization.

3.1 USPTO Bulk Data Architecture

The USPTO provides the deepest dataset but presents the most significant parsing challenges due to the evolution of its data formats over decades (Greenbook, Redbook, XML). The System utilizes the **USPTO Bulk Data Storage System (BDSS)** for backfile acquisition and the **Open Data Portal (ODP)** API for real-time updates.[10]

3.1.1 The "Redbook" Parsing Challenge

USPTO data is divided into three epochs, each requiring a specialized parser:

1. **Greenbook (Pre-2001):** Fixed-width text files with idiosyncratic headers. Extremely difficult to parse reliably.
2. **Redbook (2001-2004):** SGML-based format, often containing invalid XML characters and non-closed tags.
3. **XML (2005-Present):** Standard XML, but the Document Type Definition (DTD) changes frequently (e.g., v4.0, v4.5).

The System's "Ingestion Agent" (a Python-based microservice) uses a strategy pattern to select the correct parser based on the filename. The valid filename convention for grant files is strictly defined by regex patterns such as ^ipg\d{6}\.zip$ (Image Patent Grant, YYMMDD).[35]

Technical Implementation of the Parser:

The parser must handle "concatenated XML," where a single file contains thousands of XML root elements stacked sequentially—a format that standard XML libraries reject. We implement a stream-based splitter:

Python

```
import re
from lxml import etree

def stream_uspto_xml(file_path):
    """
    Splits concatenated USPTO XML files into individual patent XML strings.
    Handles the 'Redbook' format where multiple <?xml...?> declarations exist.
    """
    with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
        buffer =
        for line in f:
            if line.startswith('<?xml') and buffer:
                yield "".join(buffer)
                buffer =
            buffer.append(line)
        if buffer:
```

```
            yield "".join(buffer)

def parse_patent(xml_string):
    """
    Parses a single patent XML, extracting critical bibliometric fields.
    """
    try:
        root = etree.fromstring(xml_string.encode('utf-8'))
        # Extract Grant Date
        grant_date = root.xpath('//publication-reference/document-id/date/text()')
        # Extract Citations (UREF/FREF tags)
        citations = root.xpath('//references-cited/citation/patcit/document-id/doc-number/text()')
        # Extract Abstract
        abstract = root.xpath('//abstract/p/text()')
        return {
            'date': grant_date,
            'citations': citations,
            'abstract': " ".join(abstract)
        }
    except etree.XMLSyntaxError:
        # Log error for manual review or regex fallback
        return None
```

[37]

This agent normalizes all output into a unified JSON schema, regardless of the input epoch, ensuring the Forecasting Engine receives consistent data.

3.2 EPO Open Patent Services (OPS) Integration

For European data, we utilize the **EPO OPS API v3.2**. Unlike the USPTO's bulk dumps, EPO data must be retrieved via API, which introduces strict quota management (throttling).[40]

3.2.1 Throttling Middleware Strategy

The EPO OPS API imposes strict limits (e.g., requests per minute, bandwidth per week). Exceeding these results in a 403 Forbidden ban. To manage this, the System implements a **Throttling Middleware** using the python-epo-ops-client library wrapped with dogpile.cache.[41]

The middleware intercepts every request and checks the X-Throttling-Control header returned by the EPO server. If the limit is approaching (e.g., "green" changes to "yellow" or "red"), the middleware automatically sleeps the thread or queues the request for later execution.

Middleware Code Logic:

Python

```
import time
from epo_ops.middlewares import Middleware

class Throttler(Middleware):
    def process_request(self, request):
        # Check local redis counter for estimated quota
        if self.is_quota_exceeded():
            time.sleep(60) # Backoff
        return request

    def process_response(self, response):
        # Update local counter based on headers
        throttle_header = response.headers.get('X-Throttling-Control')
        if 'black' in throttle_header: # Immediate ban risk
            self.trigger_emergency_stop()
        return response
```

[40]

This ensures 24/7 ingestion without service interruption, a critical requirement for a production-grade system.

3.3 WIPO and International Data

WIPO (PCT) data is crucial for forecasting because PCT applications often represent high-value inventions intended for global protection. The System accesses this via the **WIPO IP Statistics Data Center API**. While bulk downloads are less frequent, the API allows for targeted retrieval of "International Search Reports" (ISR), which contain examiner citations—highly valuable signals for importance.[9]

---

4. Scalable Architecture: Vectors, Graphs, and Microservices

To process 100 million patents and billions of citation links, the System requires a "Big Data" architecture. A monolithic SQL database (e.g., PostgreSQL) cannot handle the graph traversal depth or the vector similarity search speed required. We adopt a **Hybrid RAG** architecture, combining a specialized Vector Database with a Graph Database.[7]

4.1 Vector Infrastructure: Milvus Distributed Cluster

For semantic search (e.g., "Find patents related to non-flammable electrolytes"), we employ **Milvus 2.0**. We select Milvus over competitors like Pinecone or Weaviate for three key

reasons:

1. **Scale:** Milvus is architected for "billion-scale" vectors, separating storage (S3/MinIO), compute (Query Nodes), and metadata (etcd). This aligns with our 100M+ document requirement.[44]
2. **Performance:** Benchmarks demonstrate Milvus achieving ~2,400 Queries Per Second (QPS) on standard datasets, significantly outperforming Weaviate (~1,500 QPS) and Chroma (~180 QPS).[46]
3. **Cost:** For a dataset of this magnitude, managed SaaS solutions (Pinecone) are cost-prohibitive. Self-hosted Milvus on AWS, utilizing Spot Instances for stateless Query Nodes, offers a ~70% reduction in Total Cost of Ownership (TCO).[47]

Cluster Configuration (AWS):

- **Query Nodes:** r6idn.24xlarge (Memory optimized for loading indices).
- **Index Type:** IVF_PQ (Product Quantization) for the best balance of RAM usage and recall, or DiskANN for SSD-resident indices if RAM costs become too high.[49]
- **Storage:** Amazon S3 for immutable segment persistence.[50]

4.2 Graph Infrastructure: Neo4j for Citation Topology

While vectors handle *similarity*, they cannot model the *flow* of knowledge. We utilize **Neo4j** to store the citation network. This allows for native execution of graph algorithms like PageRank (for importance) and Community Detection (for identifying technology clusters).[51]

Graph Schema:

- **Nodes:** Patent, Inventor, Assignee, Concept, CPC_Class.
- **Edges:** CITES (Directed), ASSIGNED_TO, INVENTED_BY, CLASSIFIED_AS.

The integration of Milvus and Neo4j enables **GraphRAG**. When a query is received, the system retrieves relevant nodes via Milvus (semantic search), then traverses the Neo4j graph to find "cousin" patents that are not textually similar but are citation-linked (structural relevance). This resolves the "vocabulary mismatch" problem in patent search.[53]

4.3 Infrastructure Sizing and Costs

For a corpus of 100M patents (approx. 2KB text each -> 200TB raw text, compressed to ~50TB vectors):

- **Vector Storage:** 100M vectors x 1536 dims (OpenAI) x 4 bytes = ~600GB index (in RAM/Disk).
- **Graph Storage:** 100M nodes + 1B edges = ~1TB graph size.

Estimated Monthly Cloud Cost (AWS):

- Milvus Cluster (3x r6i.4xlarge): ~$3,000/mo.
- Neo4j Cluster (3x r6i.4xlarge): ~$3,000/mo.
- S3 & Data Transfer: ~$500/mo.
- LLM Inference (OpenAI/Anthropic APIs): Variable, likely ~$5,000/mo at scale.
- **Total MVP Cost:** ~$11,500/month. This is significantly cheaper than the enterprise license for GetFocus, providing a sovereign asset.[55]

---

5. The "Agentic RAG" Reasoning Engine

The true differentiator of this System is the shift from static dashboards to **Agentic RAG**. In a traditional dashboard, the user must know what to ask. In an Agentic system, the user provides a high-level goal, and autonomous software agents plan and execute the research.[5]

5.1 The Workflow Orchestrator (LangGraph)

We utilize **LangGraph** to define the state machine for the research agents. The workflow follows a "Plan-Execute-Refine" loop.

Agent Roles:

1. **The Planner:** Decomposes a query like "Assess the maturity of Solid State Batteries" into sub-questions: "What is the patent volume trend?", "Who are the key assignees?", "What is the cycle time?", "Are there recent manufacturing breakthroughs?".
2. **The Retriever (Tool User):** Interfaces with Milvus and Neo4j. It decides whether to run a Vector Search (for concepts) or a Graph Query (for citation stats).
3. **The Analyst (Calculation Engine):** Runs the Python scripts for Benson-Magee TIR and Cycle Time on the retrieved dataset.
4. **The Synthesizer:** Combines the quantitative data with qualitative summaries to write the final report.[57]

5.2 Handling "Global" Queries with GraphRAG

Standard RAG fails at "Global" queries (e.g., "What are the main themes in this dataset?"). It only retrieves local chunks. Our System implements the **Microsoft GraphRAG** pattern to solve this.

Process:

1. **Community Detection:** The Neo4j graph is partitioned into communities using the Leiden algorithm.
2. **Summarization:** An LLM generates a summary for each community (e.g., "Community A: Lithium Metal Anodes," "Community B: Ceramic Electrolytes").
3. **Global Answer:** When the user asks "What are the bottlenecks?", the Agent reads the *community summaries* rather than individual patents. This allows for a comprehensive answer grounded in the entire dataset.[8]

5.3 Prompt Engineering Strategy

To ensure reliability, the Agents operate with "Chain-of-Thought" (CoT) prompting.

Example System Prompt:

"You are a Patent Analytics Agent. You have access to a Citation Graph and a Vector Index. To answer the user's question, you must first propose a research plan. Do not answer immediately. Break the problem into steps. For each step, specify which tool (Vector or Graph) is most appropriate. If you need to calculate an improvement rate, use the calculate_tir tool."

This structured approach minimizes hallucinations and ensures the mathematical models are invoked correctly.[56]

---

6. Forecasting Engine: Implementation of the S-Curve

The Forecasting Engine is the module that turns data into prediction. It implements the Fisher-Pry transform to fit the "S-curve" of technological substitution.

6.1 Algorithmic Implementation

The engine takes time-series data generated by the KPL (e.g., cumulative weighted patent counts) and fits it to the logistic function.

Python Implementation Logic:

Python

```
import numpy as np
from scipy.optimize import curve_fit

def fisher_pry_transform(f):
    """
    Linearizes the S-curve fraction f.
    f: fraction of market/theoretical limit (0 < f < 1)
    """
    # Clip to avoid log(0) errors
    f = np.clip(f, 0.001, 0.999)
    return np.log(f / (1 - f))

def fit_forecast(years, cumulative_data):
    """
    Fits the Fisher-Pry model to historical data.
    """
    # Normalize to estimate 'f'
    saturation_limit = estimate_saturation(cumulative_data)
    f = cumulative_data / saturation_limit

    # Transform
    y_transform = fisher_pry_transform(f)

    # Linear Regression: Y = alpha + beta * t
    slope, intercept = np.polyfit(years, y_transform, 1)

    # Project forward
    future_years = np.arange(years[-1], years[-1] + 10)
```

```
    future_y = intercept + slope * future_years

    # Inverse Transform to get S-curve
    future_f = 1 / (1 + np.exp(-future_y))

    return future_years, future_f * saturation_limit
```

[14]

6.2 Detecting Inflection Points

The most critical output is the **Inflection Point (t_0)**—the time when the technology hits 50% saturation and growth begins to decelerate.

- If $t_{current} < t_0$: The technology is in the exponential growth phase (Invest).
- If $t_{current} > t_0$: The technology is maturing (Harvest/Divest).

The System visualizes this with confidence intervals, highlighting the "Window of Opportunity" for R&D investment.[62]

---

7. Frontend Experience: The "God View"

The user interface (UI) serves to demystify the complex bibliometrics. It is built using **React** and **D3.js** for high-performance visualizations.

7.1 The S-Curve Dashboard

The primary view is a multi-line chart comparing competing technologies (e.g., LCD vs. OLED).

- **X-Axis:** Year.
- **Y-Axis:** Performance Metric (inferred from TIR) or Market Penetration.
- **Interaction:** Users can drag a "Scenario Slider" to adjust the assumed saturation limit, updating the forecast in real-time. This "What-if" analysis is crucial for strategic planning.[63]

7.2 The Whitespace Map

Using dimensionality reduction techniques (UMAP or t-SNE) on the Milvus vector embeddings, the System generates a 2D map of the IP landscape.

- **Clusters:** Colored dots represent existing patents.
- **Whitespace:** Empty voids between clusters represent potential innovation opportunities (e.g., a gap between "AI" and "Material Science").
- **Heatmap:** We overlay the "Cycle Time" metric as a heatmap. "Hot" (red) areas are fast-moving; "Cold" (blue) areas are stagnant. This guides users to the most active research fronts.[65]

7.3 "Chat with Patents"

A side-panel chat interface allows users to converse with the Agentic RAG.

- **User:** "Why is the forecast for Solid State Batteries delayed?"
- **System:** "Analysis of the citation graph shows a bottleneck in 'Interface Stability' (CPC H01M 10/0562). Patents in this cluster have a high Cycle Time (12 years) and low SPLC, indicating a lack of breakthrough consensus. However, a new cluster of 'Sulfide-based electrolytes' is emerging with a Cycle Time of 3 years." [1]

---

8. Quality Assurance and Backtesting Protocol

A forecasting system is only as good as its proven accuracy. We implement a rigorous **Backtesting Loop** to validate the model's predictions against historical outcomes.

8.1 The "LCD vs. CRT" Validation Case Study

To prove the system works, we replicate the historical substitution of Cathode Ray Tubes (CRT) by Liquid Crystal Displays (LCD).

1. **Data Cutoff:** We restrict the System's data access to patents granted *before 2005*.
2. **Prediction:** The System must generate an S-curve forecast for LCD adoption.
3. **Validation:** We compare the 2005 forecast against the actual market data from 2005-2015.
    - *Success Metric:* The predicted inflection point must be within ± 1.5 years of the actual inflection point.
    - *Prior Research:* Historical analysis shows LCD patents exhibited a sharp decrease in Cycle Time and increase in TIR around 2002, preceding market dominance. The System must replicate this signal.[4]

8.2 Forward Citation Prediction

We test the KPL's ability to identify "important" patents early.

1. **Training:** Train the model on patents from 2010-2015.
2. **Task:** Predict the top 5% most-cited patents of 2020.
3. **Validation:** Calculate the **Normalized Discounted Cumulative Gain (NDCG)** of the ranking.
    - *Success Metric:* NDCG > 0.8. This proves the "Immediacy + Importance" formula correctly identifies winners.[18]

---

9. Strategic Roadmap: MVP to Scale

Phase 1: The Foundation (Months 1-3)

- **Objective:** Ingest USPTO Redbook data and establish Milvus/Neo4j infrastructure.
- **Deliverables:**
    - Robust XML Parsers for all USPTO epochs.

    - Milvus Cluster on AWS.
    - Basic Vector Search API (FastAPI).

Phase 2: The Scientific Core (Months 4-6)

- **Objective:** Implement Benson-Magee/Trancik metrics.
- **Deliverables:**
    - Cycle Time Calculation Agent.
    - SPLC Algorithm optimization (subgraph execution).
    - Fisher-Pry Forecasting Module.

Phase 3: The Agentic Layer (Months 7-9)

- **Objective:** Deploy GraphRAG and LangGraph Agents.
- **Deliverables:**
    - Agent Orchestrator with Planner/Retriever roles.
    - Community Detection in Neo4j for Global Search.
    - "Chat with Patents" MVP.

Phase 4: Global Scale & Polish (Months 10-12)

- **Objective:** Integrate EPO/WIPO and finalize UI.
- **Deliverables:**
    - EPO OPS Throttling Middleware.
    - Global patent coverage.
    - Production-grade React Dashboard with interactive S-curves.

---

10. Conclusion

This Technical Blueprint outlines a path to building a world-class technology forecasting platform that rivals GetFocus/Odin. By anchoring the system in the proven bibliometric science of Benson, Magee, and Trancik, and operationalizing these metrics through a state-of-the-art **Agentic RAG** architecture, we create a tool that offers deep, actionable foresight. The combination of **Milvus** for scale, **Neo4j** for structure, and **LangGraph** for reasoning creates a "Sovereign Brain" capable of identifying the technologies of tomorrow. The engineering challenges—particularly in data parsing and graph scalability—are significant, but the roadmap provided herein offers a clear, validated path to execution.
System Status: Architecture Approved. Ready for Phase 1 Implementation.
Citations:

[1]

Referenzen

1. About us - GetFocus, Zugriff am Januar 1, 2026,

https://www.getfocus.eu/about-us
2. State of LDES - A Report on the Future of Long-Duration Energy Storage, Zugriff am Januar 1, 2026, https://cdn.prod.website-files.com/6891f1b26aafe2063f6f97f7/68c85afc7dfcb52d240ffb44_GetFocus%20Report%20-%20State%20of%20LDES.pdf
3. How it works - GetFocus, Zugriff am Januar 1, 2026, https://www.getfocus.eu/how-it-works
4. A technology replacement model with variable market potential — An empirical study of CRT and LCD TV - ResearchGate, Zugriff am Januar 1, 2026, https://www.researchgate.net/publication/223273119_A_technology_replacement_model_with_variable_market_potential_-_An_empirical_study_of_CRT_and_LCD_TV
5. What Is Agentic RAG? - Salesforce, Zugriff am Januar 1, 2026, https://www.salesforce.com/agentforce/agentic-rag/
6. Bringing agentic Retrieval Augmented Generation to Amazon Q Business - AWS, Zugriff am Januar 1, 2026, https://aws.amazon.com/blogs/machine-learning/bringing-agentic-retrieval-augmented-generation-to-amazon-q-business/
7. RAG Tutorial: How to Build a RAG System on a Knowledge Graph - Neo4j, Zugriff am Januar 1, 2026, https://neo4j.com/blog/developer/rag-tutorial/
8. GraphRAG: Unlocking LLM discovery on narrative private data - Microsoft Research, Zugriff am Januar 1, 2026, https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/
9. Chapter 7 Databases | The WIPO Manual on Open Source Patent Analytics (2nd edition), Zugriff am Januar 1, 2026, https://wipo-analytics.github.io/manual/databases.html
10. Getting Started | Open Data Portal - USPTO, Zugriff am Januar 1, 2026, https://data.uspto.gov/apis/getting-started
11. Tracing Technological Development Trajectories: A Genetic Knowledge Persistence-Based Main Path Approach - DSpace@MIT, Zugriff am Januar 1, 2026, https://dspace.mit.edu/bitstream/handle/1721.1/110026/Park-2017-Tracing%20Technological%20Development%20Tr.pdf?sequence=1&isAllowed=y
12. Main path analysis - Wikipedia, Zugriff am Januar 1, 2026, https://en.wikipedia.org/wiki/Main_path_analysis
13. Forecasting OLED TV Technology Using Bibliometrics and Fisher-Pry Diffusion Model - PDXScholar, Zugriff am Januar 1, 2026, https://pdxscholar.library.pdx.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1057&context=etm_fac
14. A Clarification of the Labeling of “Fisher-Pry” Transform Figures, Zugriff am Januar 1, 2026, https://phe.rockefeller.edu/LogletLab/FP_Clarification.pdf
15. Dashboard Builder – Opturo Inc, Zugriff am Januar 1, 2026, https://opturo.com/odin-features/dashboard-builder/
16. Technological improvement rate estimates for all technologies: Use of patent data and an extended domain description - DSpace@MIT, Zugriff am Januar 1, 2026,

https://dspace.mit.edu/bitstream/handle/1721.1/132865/1263347052-MIT.pdf?sequence=1&isAllowed=y
17. Determinants of technology improvement - Trancik Lab - MIT, Zugriff am Januar 1, 2026, https://trancik.mit.edu/determinants-of-the-rate-of-technological-improvement/
18. Quantitative Determination of Technological Improvement from Patent Data - DSpace@MIT, Zugriff am Januar 1, 2026, https://dspace.mit.edu/bitstream/handle/1721.1/96760/Magee_Quantitative%20determinatino.pdf?sequence=1&isAllowed=y
19. Quantitative Determination of Technological Improvement from ..., Zugriff am Januar 1, 2026, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0121635
20. Quantitative Determination of Technological Improvement from Patent Data - PMC - NIH, Zugriff am Januar 1, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC4398537/
21. Patents forecast technological change - ScienceDaily, Zugriff am Januar 1, 2026, https://www.sciencedaily.com/releases/2015/04/150415155329.htm
22. A Patent and Trademark Office Review - USPTO, Zugriff am Januar 1, 2026, https://www.uspto.gov/sites/default/files/about/stratplan/ar/1997annualreport.pdf
23. On Predictive Patent Valuation: Forecasting Patent Citations and Their Types - AAAI Publications, Zugriff am Januar 1, 2026, https://ojs.aaai.org/index.php/AAAI/article/view/10722/10581
24. 16 Patent Query Formulation by Synthesizing Multiple Sources of Relevance Evidence - Idiap Research Institute — EN, Zugriff am Januar 1, 2026, https://www.idiap.ch/~pmahdabi/papers/a16-mahdabi.pdf
25. Aligning innovation to business strategy : combining cross-industry and longitudinal perspectives on strategic alignment in lead - WUR eDepot, Zugriff am Januar 1, 2026, https://edepot.wur.nl/44228
26. Technological Convergence and Innovation Pathways in Sustainable Logistics Systems: An Integrated Graph Neural Network and Main Path Analysis - MDPI, Zugriff am Januar 1, 2026, https://www.mdpi.com/2071-1050/17/23/10507
27. An integrated approach for main path analysis: Development of the Hirsch index as an example | Semantic Scholar, Zugriff am Januar 1, 2026, https://www.semanticscholar.org/paper/An-integrated-approach-for-main-path-analysis%3A-of-Liu-Lu/7fd6b19c5aae90c3f69fa334135918d930bfcd12
28. 32 A Network Analysis Approach to Intellectual Property Research - Oxford Academic, Zugriff am Januar 1, 2026, https://academic.oup.com/book/41122/chapter/350442943
29. Role of design complexity in technology improvement | PNAS, Zugriff am Januar 1, 2026, https://www.pnas.org/doi/10.1073/pnas.1017298108
30. Major transitions in information technology | Philosophical Transactions of the Royal Society B, Zugriff am Januar 1, 2026, https://royalsocietypublishing.org/rstb/article/371/1701/20150450/22947/Major-transitions-in-information-technologyIT
31. Evolving complexity: how tinkering shapes cells, software and ecological

networks - PMC, Zugriff am Januar 1, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC7061959/
32. Forecasting technology success based on patent data, Zugriff am Januar 1, 2026, http://www.nhu.edu.tw/~lbhung/10401TMpapers/Forecasting%20technology%20success%20based%20on%20patent%20data.pdf
33. OLED TV technology forecasting using technology mining and the Fisher-Pry diffusion model - ResearchGate, Zugriff am Januar 1, 2026, https://www.researchgate.net/publication/297527213_OLED_TV_technology_forecasting_using_technology_mining_and_the_Fisher-Pry_diffusion_model
34. Open Data Portal - USPTO, Zugriff am Januar 1, 2026, https://data.uspto.gov/
35. Update regular expression pattern for uspto (US Patent Numbers) · Issue #253 · biopragmatics/bioregistry - GitHub, Zugriff am Januar 1, 2026, https://github.com/biopragmatics/bioregistry/issues/253
36. Bulk Data Directory | Open Data Portal - USPTO, Zugriff am Januar 1, 2026, https://data.uspto.gov/bulkdata
37. TamerKhraisha/uspto-patent-data-parser: A python tool for reading, parsing and finding patent using the United States Patent and Trademark (USPTO) Bulk Data Storage System. - GitHub, Zugriff am Januar 1, 2026, https://github.com/TamerKhraisha/uspto-patent-data-parser
38. lettergram/parse-uspto-xml - GitHub, Zugriff am Januar 1, 2026, https://github.com/lettergram/parse-uspto-xml
39. Federal trademark searching: Field tag searching with regular expressions - USPTO, Zugriff am Januar 1, 2026, https://www.uspto.gov/sites/default/files/documents/TM-FieldTagSearching-RegEx-slides-20240212.pdf
40. EPO Developer Portal: Home Page, Zugriff am Januar 1, 2026, https://developers.epo.org/
41. python-epo-ops-client/epo_ops/api.py at main - GitHub, Zugriff am Januar 1, 2026, https://github.com/ip-tools/python-epo-ops-client/blob/master/epo_ops/api.py
42. python-epo-ops-client 0.1.1 - PyPI, Zugriff am Januar 1, 2026, https://pypi.org/project/python-epo-ops-client/0.1.1/
43. API Catalog for Intellectual Property - WIPO, Zugriff am Januar 1, 2026, https://www.wipo.int/en/web/standards/ip-api-catalog/index
44. How does Milvus compare to other vector databases like Pinecone or Weaviate?, Zugriff am Januar 1, 2026, https://milvus.io/ai-quick-reference/how-does-milvus-compare-to-other-vector-databases-like-pinecone-or-weaviate
45. Introducing AISAQ in Milvus: Billion-Scale Vector Search Just Got 3200× Cheaper on Memory, Zugriff am Januar 1, 2026, https://milvus.io/blog/introducing-aisaq-in-milvus-billion-scale-vector-search-got-3200-cheaper-on-memory.md
46. I Tested 5 Vector Databases at Scale. Here's What Actually Matters. | by Preksha Dewoolkar, Zugriff am Januar 1, 2026, https://medium.com/@officialpreksha2166/i-tested-5-vector-databases-at-scale

-heres-what-actually-matters-93fb997e21b0
47. Best Vector Databases in 2025: A Complete Comparison Guide - Firecrawl, Zugriff am Januar 1, 2026, https://www.firecrawl.dev/blog/best-vector-databases-2025
48. What is the pricing model for serverless services? - Milvus, Zugriff am Januar 1, 2026, https://milvus.io/ai-quick-reference/what-is-the-pricing-model-for-serverless-services
49. Vector Database Comparison for AI Developers - Medium, Zugriff am Januar 1, 2026, https://medium.com/@felix-pappe/vector-database-comparison-for-ai-developers-90aeb3d79caf
50. How to Deploy the Open-Source Milvus Vector Database on Amazon EKS, Zugriff am Januar 1, 2026, https://milvus.io/blog/how-to-deploy-open-source-milvus-vector-database-on-amazon-eks.md
51. NODES 2024 - Building Knowledge Graphs with LLMs from USPTO Patent Data - YouTube, Zugriff am Januar 1, 2026, https://www.youtube.com/watch?v=Px1q37clDKE
52. Knowledge Graph vs. Vector RAG: Optimization & Analysis - Neo4j, Zugriff am Januar 1, 2026, https://neo4j.com/blog/developer/knowledge-graph-vs-vector-rag/
53. GraphRAG vs Traditional RAG: Knowledge Graphs for Accurate, Enhanced RAG Applications | by Tahir | Medium, Zugriff am Januar 1, 2026, https://medium.com/@tahirbalarabe2/graphrag-vs-traditional-rag-knowledge-graphs-for-accurate-enhanced-rag-applications-2cc4f6f9f4b4
54. Graph RAG vs. Classical RAG: A Comparative Analysis - ELEKS, Zugriff am Januar 1, 2026, https://eleks.com/research/graph-rag-vs-classical-rag-analysis/
55. AWS Marketplace: Milvus DB: AI-Ready Vector Database Environment - Amazon.com, Zugriff am Januar 1, 2026, https://aws.amazon.com/marketplace/pp/prodview-naacryhk47d6e
56. RAG vs Fine-tuning vs Long Context: When to Use What (And Why Most Teams Get It Wrong) | by Preksha Dewoolkar | Dec, 2025 | Medium, Zugriff am Januar 1, 2026, https://medium.com/@officialpreksha2166/rag-vs-fine-tuning-vs-long-context-when-to-use-what-and-why-most-teams-get-it-wrong-388cc446ff3c
57. Agentic Workflow for Competitive Intelligence: How AI-Powered Systems Transform Market Tracking | by Sarah Morino | Nov, 2025, Zugriff am Januar 1, 2026, https://ai.plainenglish.io/agentic-workflow-for-competitive-intelligence-how-ai-powered-systems-transform-market-tracking-587b65496611
58. Agentic RAG Workflow: The Future of AI Information Retrieval | by Munsif Raza - Medium, Zugriff am Januar 1, 2026, https://medium.com/@munsifrazaofficial/agentic-rag-workflow-the-future-of-ai-information-retrieval-ecdbf2f6695d

59. Intro to GraphRAG, Zugriff am Januar 1, 2026, https://graphrag.com/concepts/intro-to-graphrag/
60. GraphRAG Costs Explained: What You Need to Know | Microsoft Community Hub, Zugriff am Januar 1, 2026, https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/graphrag-costs-explained-what-you-need-to-know/4207978
61. Modeling diffusion of multi-generational LCD TVs while considering generation-specific price effects and consumer behaviors, Zugriff am Januar 1, 2026, https://ir.lib.nycu.edu.tw/bitstream/11536/22997/1/000326213400005.pdf
62. The Use of S Curves in Technology Forecasting and its Application On 3D TV Technology - Zenodo, Zugriff am Januar 1, 2026, https://zenodo.org/record/1080624/files/13173.pdf
63. S-Curve Forecasting Tool for Construction Projects - Anterra Technology, Zugriff am Januar 1, 2026, https://anterratech.com/modules/s-curve-forecasting/
64. Project Tracking Unleashed: Build S-Curves with Power BI (Part 1) - Smart Frames UI, Zugriff am Januar 1, 2026, https://smart-frames.co.uk/2025/02/10/create-an-s-curve-for-project-tracking-in-power-bi-part-1/
65. Development and Implementation of Trajectory Optimization Technologies for Cranial Stereotactic Radiation Therapy - ADVAN-KT, Zugriff am Januar 1, 2026, https://advan-kt.com/doc/medical2.pdf
66. On Predictive Patent Valuation: Forecasting Patent Citations and Their Types | Request PDF, Zugriff am Januar 1, 2026, https://www.researchgate.net/publication/361529872_On_Predictive_Patent_Valuation_Forecasting_Patent_Citations_and_Their_Types