



The Smart Decision Group

# Decision Engines 101

Cross-industry guide to automating decisions with rules, analytics, and AI

Analytics, Automation, Advantage

2025 White Paper

## 1. What is a Decision Engine

A Decision Engine evaluates input data using business rules, scorecards, statistical models, or AI/ML algorithms to produce outcomes such as approve, decline, price, rank, or personalize. It ensures transparent, consistent, and scalable decision-making across products, channels, and business units.

### Key Properties:

- Deterministic when required; probabilistic when beneficial.
- Transparent, auditable, evidence-based decisions.
- Low-latency APIs for real-time use; batch processing for analytics and operations.
- Full versioning of logic, models, and configuration.

## 2. Comparison: Manual Decisions vs Coded Rules vs Decision Engines

To understand the value of a modern decision engine, it is helpful to compare it with traditional manual decisioning and hard-coded rules systems.

### 1. Manual Decisioning

- Humans evaluate documents, policies, and customer data manually.
- Highly inconsistent — outcomes vary across agents, days, and branches.
- Slow: minutes to days depending on workload.
- Difficult to audit — decisions rely on judgement notes and emails.
- Scaling requires more staff, increasing cost-per-decision.
- High risk of error, bias, fraud, and policy drift.

### 2. Hard-Coded Rules Systems

- Rules are embedded directly inside application code (Java, C#, Python).
- Faster and more consistent than manual processes.
- Changes require developers → long change cycles, release dependencies.
- Poor transparency: business users cannot see or edit rules.
- Versioning is difficult; rollback requires code deployment.
- Limited ability to explain outcomes or provide regulatory reason codes.
- Adding ML models or statistical scorecards is complex and fragile.

### 3. Decision Engine (Rules + Models Platform)

- Centralised platform for business rules, scorecards, and AI models.
- Changes can be made quickly by business analysts without code deployments.
- Full transparency and audit of every change, version, and outcome.
- Reason codes generated automatically for compliance.
- Scales horizontally via APIs; low-latency decisions in milliseconds.
- Integrates smoothly with bureaus, CRM, ERP, KYC, fraud, and data sources.
- Supports experiments, A/B tests, drift monitoring, and safe rollouts.

### Summary:

Manual systems lack scale and consistency. Hard-coded rules systems provide scale but lack agility and visibility. Decision engines combine speed, transparency, explainability, and control — enabling real-time, governed, data-driven decisions at enterprise scale.

Dimension	Manual Decisions	Coded Rules System	Decision Engine
Consistency	Low: varies per agent	Medium: consistent code coded	High: controlled centrally
Change Speed	Slow	Slow: requires developers	Fast: business-editable

<b>Auditability</b>	Weak	Medium: code history only	Strong, full versioning
<b>Scalability</b>	Poor	Good	Excellent
<b>Explainability</b>	Low	Low-Medium	High: reason codes

Table 1 — Comparison Table: Manual vs Coded Rules vs Decision Engine

### 3. Quantified Example Improvements

#### Time-to-decision:

- Manual: 30 minutes to days
- Coded rules: 2–10 seconds
- Decision engine: 50–200 ms

#### Consistency (variance in outcomes for identical applicants):

- Manual:  $\pm 20\text{--}40\%$
- Coded rules:  $\pm 5\text{--}10\%$
- Decision engine:  $<1\%$  (fully deterministic unless models introduce probabilistic elements)

#### Change cycle duration:

- Manual SOP updates: weeks to months
- Coded rules: 2–6 weeks per release
- Decision engine: minutes to hours

#### Error rate / compliance exceptions:

- Manual: 3–7% typical
- Coded rules: 1–3%
- Decision engine:  $<0.5\%$

### 4. Reference Architecture

A decision engine follows a layered architecture that separates logic from integration and data.

#### Layers:

- Input Layer — Data from CRM, ERP, LMS, bureaus, devices, internal systems, and contextual signals.
- Rules & Models — Business rules, constraints, scorecards, ML models, optimisation logic.
- Decision Layer — Combines evidence, calculates scores, and produces reason codes.
- Integration Layer — REST, GraphQL, events, and data pipelines into warehouses.
- Monitoring — Drift, fairness, stability, latency, performance telemetry, and audit.

### 5. Strategic Benefits

- Consistency and accuracy across all channels.
- Reduced time-to-decision and lower cost-per-decision.
- Full auditability and regulatory alignment.
- Safer, faster change cycles with structured governance.

### 6. Real-World Applications

- Lending: origination, credit underwriting, pricing.
- Fraud: KYC, sanctions, device risk, behavioural anomalies.
- Insurance: claims triage, adjudication, agent scoring.
- Telecom: churn prevention, retention offers, fair-use controls.
- Retail: promotions, personalisation, recommender systems.
- Public sector: grants, tenders, permits, benefits eligibility.

## 7. Common Integrations

- CRM – Customer profiles and interactions for personalisation and pre-fill.
- ERP – Finance, inventory, and operational data for affordability and risk.
- LMS – Contract setup, schedule generation, collections and downstream updates.
- Data Sources – Own or third party data sources
- Front End systems – System that allows applicant data to be captured, and displays results.

## 8. Data Exchange

- Real-time API's for in-journey decisions.
- Batch or streaming for analytics and operational stores

## 9. Anti-Fraud & Tender Vetting (South Africa)

Decision engines enable automated verification, conflict checks, and fraud detection, reducing identity and tender fraud.

## 10. Examples

- Document liveness, OCR checks, tamper detection.
- Company register cross-checks; director linkage; duplicate banking flags.
- Device fingerprints, geolocation anomalies.
- Risk-based referrals with full audit logs.

## 11. Change Management & Governance

A strong governance framework ensures safe updates to rules, models, and policies.

## 12. Components:

- Maker-checker approvals and role-based access.
- Sandbox testing and simulation.
- A/B experiments and controlled rollouts.
- Monitoring of reason codes for fairness and compliance drift.

## 13. Data & Feature Engineering

Feature engineering improves model quality and interpretability.

## 14. Key Practices:

- Compute interpretable variables: utilisation, affordability ratios, behavioural aggregates.
- Monitor outliers, drift, and missingness.
- For high-frequency markets, compute rolling averages (3/6/9/12) and % deviation signals.
- Prefer interpretable statistics over opaque candlestick labels.

## 15. Orchestration: Combining Rules & Models

Effective decisioning blends deterministic rules with predictive models.

- Guardrail rules enforce mandatory constraints.
- Scorecards compute probabilities or risk levels.
- Pricing models optimise outcomes.
- Models feed into refer/decline logic and explainable outcomes.

## 16. Implementation Playbook

1. Define outcomes, constraints, KPIs, and reason codes.

2. Map data sources and agree integration contracts.
3. Build v1 rules and baseline models; implement CI/CD.
4. Pilot with A/B experiments and measure performance.
5. Iterate using monitored insights and change requests.

## 17. Glossary

Decision Engine — System that transforms data, rules, and models into a decision.

CRM — Customer Relationship Management platform.

ERP — Enterprise Resource Planning platform.

LMS — Loan Management System.

Scorecard — Predictive model producing a numeric risk score.

Reason Code — Human-readable explanation of a decision.

Maker-Checker — Two-person approval workflow.

### **Contact Details**

The Smart Decision Group (TSDG)

Email: [eugeneehl@outlook.com](mailto:eugeneehl@outlook.com)

Web: [www.tsdg.co.za](http://www.tsdg.co.za)