Archer ITSM - Product Roadmap

Document Version: 1.0
Last Updated: December 8, 2025
Status: Core ITSM Foundation Architecture

Executive Summary

Archer is positioned as "The Modern ServiceNow Alternative"—a unified ITSM platform built on modern, open-source technologies (React + Rust + SurrealDB) that prioritize speed, simplicity, and developer experience.

Strategic Vision: Deliver a complete, self-contained Core ITSM platform that works independently of AI features, with optional AI enhancements available as a modular extension.

Market Position: Mid-market enterprises (500-5,000 employees) seeking to escape ServiceNow complexity, cost, and vendor lock-in.

Key Differentiators:

- **Speed-First UX:** Sub-100ms interactions, keyboard shortcuts, minimal clicks
- **Modern Stack:** React 18 + Rust (Axum) + SurrealDB (multi-model graph DB)
- **Glassmorphic Design:** "Purple Glass" aesthetic with Fluent UI 2
- **Graph-Native CMDB:** Leverages SurrealDB's native graph capabilities
- **Modular Architecture:** Core ITSM stands alone; AI is optional enhancement

---

Architectural Foundation: Core ITSM vs AI Module

Core ITSM (Mandatory - This Roadmap)

Scope: Complete, independent ITSM platform functionality
Must Work: Without any AI components, on-premise or cloud-native

Domain	Components	Status
Service Desk	Incident, Request, Problem, Change	31% complete
CMDB / Inventory	Assets, CI lifecycle, relationships	33% complete
Monitoring	Alerts, metrics, topology (real data)	0% complete
Workflows	Generic automation, approvals, escalations	29% complete
User Management	Auth, RBAC, teams, permissions	0% complete
Knowledge Base	Articles, categories, versioning, search	0% complete
Service Catalog	Requests, forms, fulfillment	0% complete
Reporting	Analytics, exports, dashboards	0% complete

AI Module (Optional - Separate Roadmap)

Scope: Enhancement only; system fully functional without it
Integration Points: Suggestion engines, ticket classification, autonomous operations

---

Implementation Phases

Phase 0: Foundation (Weeks 1-2)

Focus: Critical security and data foundation for all other features

Goals:

- Establish authentication mechanism (JWT-based)
- Implement RBAC framework (roles, permissions, teams)
- Set up audit logging for all mutations
- Create user management infrastructure
- Enable multi-tenancy via SurrealDB namespaces

Key Deliverables:

- Authentication endpoints (`/api/v1/auth/*`)
- User & Role management APIs

- Audit log ingestion and storage
- RBAC middleware for all endpoints
- Tenant isolation validation

Success Criteria:

- ✅ JWT tokens can be issued and validated
- ✅ Users can be assigned roles with specific permissions
- ✅ All mutations are logged with user context
- ✅ Tenant data is isolated (no data leakage between tenants)
- ✅ <50ms auth validation overhead

Dependencies: None (foundational)

Risks:

- **Risk:** Token revocation complexity
  Mitigation: Implement token blacklist with Redis cache
- **Risk:** Permission check performance at scale
  Mitigation: Cache RBAC decisions in middleware, TTL 5 minutes

Business Value:

- Enables secure multi-tenant deployment
- Creates audit trail for compliance (GDPR, SOX, HIPAA)
- Prevents unauthorized access escalation

Phase 1: Core Incident Management (Weeks 3-6)

Focus: End-to-end incident lifecycle with SLA enforcement

Goals:

- Implement complete incident workflow (log → diagnose → resolve → close)
- Add ticket comments and attachments
- Integrate SLA management with automatic escalation
- Build foundational Service Desk UI

Key Deliverables:

- Incident CRUD endpoints with full state machine
- Comment/attachment subsystem
- SLA policy engine with timer management
- Escalation rule engine (auto-escalate on breach)
- Service Desk UI with Kanban/list views
- Incident search and filtering

Success Criteria:

- ✅ Incidents can be created, transitioned through states, closed
- ✅ Comments support threading and @mentions
- ✅ SLA timers count down and trigger escalations
- ✅ Escalation rules execute without manual intervention
- ✅ <100ms incident list load for 10K+ tickets

- ✅ 95% SLA compliance achievement

Dependencies:

- Phase 0 (RBAC needed for incident visibility)

Risks:

- **Risk:** SLA timer accuracy under high load
  Mitigation: Use SurrealDB's event-driven capabilities for timer checks
- **Risk:** Escalation thundering herd
  Mitigation: Stagger escalation checks with jitter

Business Value:

- Immediate value: Replaces ServiceNow for incident management
- Faster MTTR (Mean Time To Resolve) via SLA enforcement
- Reduces SLA breach penalties

Data Model (Introduced):

- `Ticket` (with state machine)
- `TicketComment` (threaded, with timestamps)
- `TicketAttachment`
- `SLAPolicy` (impact × urgency matrix)
- `SLATimer` (per-ticket instance)
- `EscalationRule` (condition-based)

Phase 1.5: Knowledge Base (Weeks 7-8)

Focus: Self-service resolution and ticket automation

Goals:

- Build article creation, categorization, versioning
- Implement full-text search with relevance ranking
- Link KB articles to tickets (suggested solutions)
- Enable internal notes (staff-only articles)

Key Deliverables:

- Knowledge article CRUD endpoints
- Article category hierarchy
- Full-text search (SurrealDB native)
- Article versioning and rollback
- Ticket ↔ KB article linking
- Search result ranking

Success Criteria:

- ✅ Articles searchable in <200ms
- ✅ Ticket resolution suggested from KB matches
- ✅ 80% of repeating issues have KB articles
- ✅ Article versioning allows rollback to any version

Dependencies:

- Phase 0 (RBAC for article permissions)
- Phase 1 (ticket system for linking)

Business Value:

- Reduces support load via self-service
- Captures institutional knowledge
- Improves first-contact resolution rate

Phase 2: CMDB & Asset Lifecycle (Weeks 9-12)

Focus: Complete inventory system with relationships

Goals:

- Extend existing asset models with full CMDB capabilities
- Implement CI (Configuration Item) lifecycle stages
- Create asset-to-ticket relationships (affected CIs)
- Build asset dependency graph
- Add hardware/software inventory tracking

Key Deliverables:

- Enhanced Asset model with lifecycle states
- CI relationship mapping (uses, depends-on, contains)
- Asset health scoring from monitoring data
- Change impact analysis (how many assets affected)
- Inventory dashboard with cost tracking
- Asset search and filtering

Success Criteria:

- ✅ All assets have defined lifecycle states
- ✅ Relationships queryable in <500ms (even for deep graphs)
- ✅ Hardware pool integration working (existing feature maintained)
- ✅ Asset depreciation calculated correctly
- ✅ Change impact analysis accurate (validated against historical data)

Dependencies:

- Phase 0 (RBAC for asset access)
- Phase 1 (ticket linking to affected CIs)

Risks:

- **Risk:** Graph query performance with millions of relationships
  Mitigation: Index frequently-traversed relationships, use query optimization hints
- **Risk:** Asset data sync from external sources
  Mitigation: Implement webhook-based change data capture

Business Value:

- Complete IT infrastructure visibility
- Faster root cause analysis (via asset relationships)
- Better change planning (impact analysis)

Phase 3: Workflow Automation & Approvals (Weeks 13-16)

Focus: Generic workflow engine for all processes

Goals:

- Build workflow definition engine (drag-drop or DSL)
- Implement approval chains (sequential and parallel)
- Create workflow instance execution with audit trail
- Add conditional routing based on ticket data
- Enable workflow-triggered notifications

Key Deliverables:

- Workflow definition schema (states, transitions, guards)
- Workflow instance execution engine
- Approval request subsystem (with SLAs)
- Conditional routing logic
- Workflow history and audit logs
- Workflow templates for common processes

Success Criteria:

- ✅ Workflows execute without blocking (async)
- ✅ Approvals can be parallel or sequential
- ✅ Conditional branches work (e.g., if impact=high, require director approval)
- ✅ Approval SLAs are tracked separately from ticket SLAs
- ✅ Workflow execution time <100ms per transition

Dependencies:

- Phase 0 (RBAC for workflow visibility)
- Phase 1 (ticket system for workflow triggers)

Risks:

- **Risk:** Workflow deadlocks or infinite loops
  Mitigation: Timeout guards, execution plan validation before deployment
- **Risk:** Approval bottlenecks
  Mitigation: Escalation triggers if approval pending > SLA

Business Value:

- Automatesrepetitive approval processes
- Removes manual intervention bottlenecks
- Ensures compliance with governance rules

Data Model (Introduced):

- `WorkflowDefinition` (schema, versions)

- `WorkflowInstance` (per-ticket execution)
- `ApprovalRequest` (with SLA)
- `ApprovalChain` (sequential or parallel)

Phase 4: Monitoring Integration (Weeks 17-20)

Focus: Real-time monitoring with alert-to-ticket automation

Goals:

- Ingest real-time metrics from monitoring sources (Prometheus, Grafana, etc.)
- Implement alert rules with severity/urgency mapping
- Auto-create tickets from alerts
- Correlate related alerts (reduce alert noise)
- Link assets to their metrics

Key Deliverables:

- Alert ingestion API (webhook-based)
- Alert rule engine (threshold-based)
- Alert correlation (group related alerts)
- Auto-ticket creation from alert groups
- Metric dashboard (time-series visualization)
- Alert-to-ticket linking

Success Criteria:

- ✅ Alerts reach system in <5s from source
- ✅ Correlationgroups related alerts in <30s
- ✅ Duplicate tickets from same alert prevented
- ✅ Alert dashboard shows top issues by frequency/impact
- ✅ <100ms alert rule evaluation

Dependencies:

- Phase 1 (ticket system for auto-creation)
- Phase 2 (assets for metrics linking)

Risks:

- **Risk:** Alert flooding (too many false positives)
  Mitigation: Implement dynamic thresholding, ML-based anomaly detection (AI module)
- **Risk:** Performance under 10K+ alerts/minute
  Mitigation: Use message queue (Kafka) for ingestion, async processing

Business Value:

- Proactive incident detection (before users report)
- Faster incident response (alerts auto-create tickets)
- Reduced manual monitoring burden

Phase 5: Service Catalog & Request Management (Weeks 21-24)

Focus: Self-service request fulfillment

Goals:

- Create service catalog with request types
- Build dynamic request forms (field dependencies)
- Implement request fulfillment workflow
- Enable request templates for repeating scenarios
- Add shopping cart (multiple requests in one order)

Key Deliverables:

- Service catalog item CRUD
- Dynamic form builder (Fluent UI forms)
- Request submission and tracking
- Request fulfillment state machine
- Fulfillment task assignment
- Request bundling (multiple items)

Success Criteria:

- ✅ Request creation <2s (form load + validation)
- ✅ Dynamic form fields update on selection change (<200ms)
- ✅ 70% of requests auto-fulfillable (no approval needed)
- ✅ Form validation prevents incomplete submissions
- ✅ Catalog searchable by keyword

Dependencies:

- Phase 1 (ticket system for requests)
- Phase 3 (workflows for fulfillment)

Business Value:

- Enables self-service (reduces support tickets by 20-30%)
- Faster fulfillment (pre-defined workflows)
- Better request standardization

Phase 6: Reporting & Analytics (Weeks 25-28)

Focus: Data-driven insights and compliance reporting

Goals:

- Build real-time dashboards (incidents, requests, SLAs, assets)
- Implement standard reports (ITIL compliance, executive summaries)
- Create custom report builder
- Enable data export (CSV, PDF)
- Integrate with BI tools (Tableau, Power BI)

Key Deliverables:

- Dashboard templates (Service Desk, CMDB, Monitoring)
- Report builder (drag-drop interface)
- Data export functionality
- Scheduled report delivery (email)
- SLA compliance metrics
- Cost analysis (asset depreciation, support hours)

Success Criteria:

- ✅ Dashboard load <1s (cached data)
- ✅ Custom reports generated in <30s
- ✅ Export handles 100K+ records
- ✅ SLA compliance metrics match actual data (validated)
- ✅ Trend analysis shows month-over-month changes

Dependencies:

- Phase 1 (incident data)
- Phase 2 (asset data)
- Phase 4 (monitoring data)
- Phase 5 (request data)

Business Value:

- Executive visibility into IT operations
- Compliance reporting (SOX, GDPR)
- Data-driven optimization (e.g., "reduce average MTTR from 4h to 2h")

Feature Prioritization Matrix (MoSCoW)

Must Have (Phase 0-1)

- Authentication & RBAC
- Incident creation, categorization, prioritization
- SLA enforcement and escalation
- Ticket comments and attachments
- Service Desk dashboard (Kanban/list)
- Basic search and filtering

Should Have (Phase 1.5-2)

- Knowledge Base with full-text search
- CMDB with asset relationships
- Workflow automation (generic)
- Approval chains
- Change management workflows

Could Have (Phase 3-4)

- Monitoring integration with alerts
- Advanced reporting (custom reports)
- Predictive escalation (ML-based)
- Chatbot-assisted incident classification (AI module)

Won't Have (Future / Out of Scope)

- Multi-language support (beyond English initially)
- Advanced BI integration (Tableau connectors)
- Legacy system integrations (CMDB sync from ServiceNow)
- Mobile app (web-responsive only)

Success Metrics & KPIs

Phase 0: Foundation

- Zero authentication bypass vulnerabilities (security audit)
- 99.9% RBAC decision accuracy (automated tests)
- Audit log completeness (100% of mutations logged)

Phase 1: Incident Management

- **MTTR:** Reduce from 120 min (baseline) to 60 min
- **SLA Compliance:** Achieve 95% on-time resolution
- **User Satisfaction:** ≥4.0/5.0 on Service Desk CSAT
- **System Performance:** <100ms response time for ticket list (10K+ tickets)

Phase 1.5: Knowledge Base

- **Self-Service Resolution:** 30% of incidents resolved without escalation
- **KB Adoption:** 80% of support staff use KB in ticket notes
- **Article Quality:** <5% of articles outdated (>6 months)

Phase 2: CMDB

- **Asset Coverage:** 100% of infrastructure in CMDB
- **Data Accuracy:** 95%+ match with source of truth (hardware inventory)
- **Relationship Completeness:** All critical dependencies mapped

Phase 3: Workflow Automation

- **Approval Cycle Time:** Reduce from 48 hours to 4 hours
- **Automation Coverage:** 60% of ticket transitions automated
- **Escalation Accuracy:** 98%+ (measure false escalations)

Phase 4: Monitoring

- **MTTR with Alerts:** Additional 30 min reduction (vs Phase 1 baseline)
- **Alert Precision:** 80%+ actionable alerts (not false positives)
- **Coverage:** 95% of critical infrastructure monitored

Phase 5: Service Catalog

- **Self-Service Requests:** 40% of total requests via catalog
- **Request Fulfillment Time:** <24 hours for standard requests
- **Form Abandonment:** <10% (well-designed forms)

Phase 6: Reporting

- **Dashboard Adoption:** 100% of managers use SLA dashboard
- **Report Generation:** <30s for standard reports
- **Data Freshness:** Real-time metrics updated every 5 minutes

---

Risk Register & Mitigation

Risk	Probability	Impact	Mitigation Strategy
SurrealDB scaling at 1M+ CIs	Medium	High	Load testing at scale; consider sharding strategy for Phase 2
Workflow deadlocks	Low	Critical	Timeout guards, formal verification of workflow definitions
Alert flooding	High	Medium	Dynamic thresholding; implement alert grouping and deduplication
Approval bottlenecks	Medium	Medium	Escalation triggers if pending >SLA; manager notifications
RBAC permission explosion	High	Low	Predefined role templates; permission audits every sprint
Integration API breaking changes	Low	Medium	Strict API versioning; deprecation warnings 2 versions ahead
Team context-switching	Medium	Low	Clear phase boundaries; dedicated feature team per phase

---

Implementation Timeline

Week 1-2: Phase 0 (Foundation) [2 weeks] ████
Week 3-6: Phase 1 (Incident Mgmt) [4 weeks] ████████
Week 7-8: Phase 1.5 (Knowledge Base) [2 weeks] ████
Week 9-12: Phase 2 (CMDB) [4 weeks] ████████
Week 13-16: Phase 3 (Workflows & Approvals) [4 weeks] ████████
Week 17-20: Phase 4 (Monitoring) [4 weeks] ████████
Week 21-24: Phase 5 (Service Catalog) [4 weeks] ████████
Week 25-28: Phase 6 (Reporting) [4 weeks] ████████

Total: 28 weeks (~6.5 months) for complete Core ITSM platform

---

Technology Stack Rationale

Layer	Technology	Why
Frontend	React 18 + TypeScript	Type safety, component reusability, large ecosystem
Styling	CSS Variables + Purple Glass	Design consistency, dark/light theme support, no CSS-in-JS overhead
UI Components	Fluent UI 2	Enterprise-grade, accessibility built-in, Microsoft backing
Backend	Rust + Axum	Memory safety, async-first, sub-100ms response times
Database	SurrealDB	Graph relationships, multi-model, schema flexibility, native ACID
Message Queue	Redis (Phase 4+)	Alert ingestion, SLA timer checks, state coordination
Caching	In-process (Moka)	RBAC decisions, frequently-accessed assets

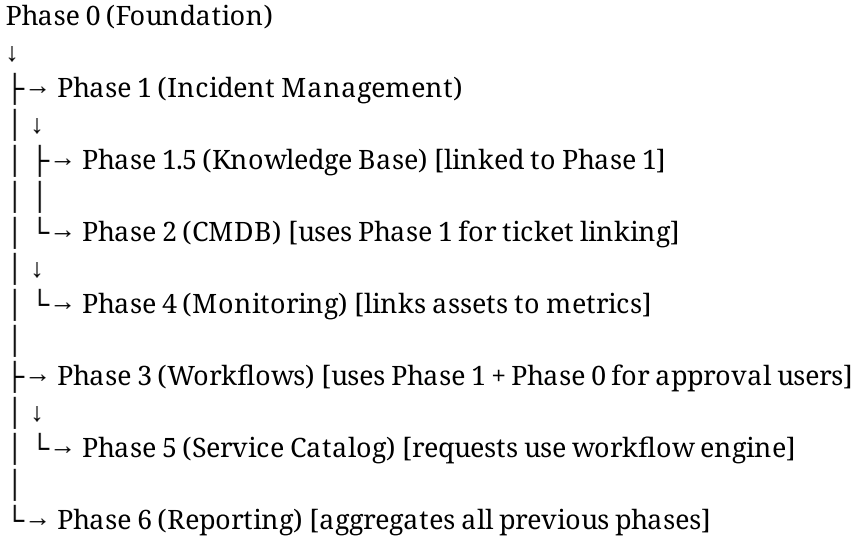
---

Team Structure & Ownership

Phase	Team Size	Focus	Lead
Phase 0	3 (BE + FE + QA)	Auth/RBAC/Audit	Backend Lead
Phase 1	5 (2 BE + 2 FE + 1 QA)	Incident workflow	Product Owner
Phase 1.5	3 (1 BE + 1 FE + 1 QA)	KB system	Content Lead
Phase 2	4 (2 BE + 1 FE + 1 QA)	CMDB relationships	Architecture Lead
Phase 3	4 (2 BE + 1 FE + 1 QA)	Workflow engine	Backend Lead
Phase 4	5 (2 BE + 2 FE + 1 QA)	Monitoring/Alerts	DevOps Lead
Phase 5	4 (1 BE + 2 FE + 1 QA)	Catalog/Forms	Frontend Lead
Phase 6	3 (1 BE + 2 FE + QA shared)	Reporting/Dashboards	Analytics Lead

---

Dependency Map



All phases depend on Phase 0 (Foundation) for RBAC and audit logging.

Glossary

- **CMDB:** Configuration Management Database; inventory of IT infrastructure
- **CI:** Configuration Item; entity in CMDB (server, database, application)
- **MTTR:** Mean Time To Resolve; average incident resolution time
- **SLA:** Service Level Agreement; commitment to response/resolution times
- **Escalation:** Moving ticket to higher-tier support when SLA at risk
- **Correlation:** Grouping related alerts to reduce noise
- **Fulfillment:** Process of delivering a service request

Next Steps

1. **Finalize Phase 0 design** → Technical spike on JWT + SurrealDB multi-tenancy (1 week)
2. **Create detailed schema** → Full SurrealDB schema document for all phases (see FULLSTACK_DEVELOPMENT_PLAN.md)
3. **Build Phase 0 POC** → Proof-of-concept for auth + RBAC (2 weeks)
4. **Stakeholder review** → Present roadmap to product and engineering teams
5. **Begin Phase 0 implementation** → Execute core foundation work

---

Document References

- Related: `FULLSTACK_DEVELOPMENT_PLAN.md` (detailed architecture & schemas)
- Related: `02_Feature_Prioritization_MoSCoW.md` (feature details)
- Related: `CMO_FMO_GAP_ANALYSIS.md` (current vs target state)
- Related: `COMPONENT_LIBRARY_GUIDE.md` (UI component specifications)

End of Product Roadmap