G1 Solution Architecture

AI-Powered Software Development Platform

Version: 1.0 **Date:** 2025-08-27 **Document Type:** Technical Architecture Specification

---

📋 Table of Contents

🏗️ Architecture Overview

System Vision

G1 is a revolutionary AI-powered software development platform that transforms business requirements into production-ready software through a network of specialized AI personas working in coordination.

Architecture Principles

- **100% Persona-Driven**: All decisions made by AI personas, zero hardcoded rules
- **Dynamic Workflow Adaptation**: Workflows designed in real-time based on project needs
- **Microservices Architecture:** Distributed, scalable, and maintainable
- **Event-Driven Communication**: Asynchronous, resilient communication patterns
- **Context-Aware Processing**: Intelligent context preservation and sharing

High-Level Architecture Diagram

System Diagram (Top-Bottom Layout)

📦 Client Layer 📦 Orchestration Layer 📦 AI Personas Layer 📦 Requirements 📦 Architecture 📦 Development 📦 Communication 📦 Infrastructure Layer 📦 Data Layer

Components:

- Web Interface
- REST API
- Command Line Interface
- Workflow Designer Persona
- Team Structure Architect
- Communication Architect
- Pure Persona Orchestrator
- Requirement Concierge
- Business Analyst
- Solution Architect
- Technical Architect
- API Designer
- Database Architect
- Developer
- Tester
- QA Specialist
- Central Knowledge Hub
- Verification Service
- Collaborative Transition Manager
- Personas Gateway
- RAG Engine
- Secrets Management
- Port Management
- (PostgreSQL)
- (Redis Cache)
- (Vector Store)
- (File Storage)

Connections:

- UI → API
- CLI → API
- API → PO
- PO → WD
- PO → TSA
- PO → CA
- WD → PG
- TSA → PG
- CA → PG
- RC → PG

🎯 Core Components

1. Pure Persona-Driven Orchestrator

Purpose: Central coordination engine that executes workflows designed entirely by AI personas.

Key Features:

- **Zero Hardcoded Rules**: All logic determined by AI personas
- **Dynamic Workflow Execution**: Adapts to any project complexity
- **Meta-Orchestration**: Uses AI personas to design workflows
- **Context Management**: Preserves requirements throughout execution
- **Quality Assurance**: Built-in verification at every step

Technical Specifications:

- **Language**: Python 3.11+
- **Framework**: AsyncIO for concurrent processing
- **API**: RESTful interfaces with OpenAPI specification
- **Scalability**: Horizontal scaling support
- **Monitoring**: Comprehensive logging and metrics

2. AI Personas Gateway

Purpose: Unified interface for all AI persona interactions with intelligent routing and context management.

Key Features:

- **Persona Management**: 22+ specialized AI personas
- **Context-Aware Routing**: Intelligent request routing
- **Response Caching**: Performance optimization
- **Load Balancing**: Distributed persona processing
- **Health Monitoring**: Real-time persona availability

Technical Specifications:

- **Framework**: FastAPI with async processing
- **Port**: 8013 (managed via port management service)
- **Authentication**: JWT-based security
- **Rate Limiting**: Configurable request throttling
- **Circuit Breaker**: Resilience patterns

3. RAG (Retrieval-Augmented Generation) Engine

Purpose: Advanced AI processing engine that combines retrieval and generation for intelligent responses.

Key Features:

- **Vector Search**: Semantic similarity matching
- **Context Integration**: Combines multiple information sources
- **LLM Integration**: OpenAI/Claude/Local models support
- **Knowledge Base**: Dynamic knowledge management
- **Response Generation**: Contextually aware responses

Technical Specifications:

- **Port**: 8003
- **Vector Database**: Pinecone/ChromaDB support
- **Embedding Models**: OpenAI/Sentence-Transformers
- **Cache Layer**: Redis for response caching
- **Security**: API key management via secrets service

🤖 AI Personas Ecosystem

Persona Categories

1. Meta-Orchestration Personas

These personas design and manage the workflow itself:

| Persona | Role | Key Responsibilities | |---------|------|---------------------| | **Workflow Designer** | SDLC Workflow Design Specialist | - Design optimal SDLC phases
- Select appropriate personas
- Create workflow sequences | | **Team Structure Architect** | Multi-Team Design Specialist | - Design optimal team structures
- Define team boundaries
- Plan coordination strategies | | **Communication Architect** | Communication Strategy Designer | - Design communication patterns
- Prevent anti-patterns
- Select communication personas |

2. SDLC Core Personas

Traditional software development roles as AI personas:

System Diagram (Left-Right Layout)

📦 Requirements Phase 📦 Architecture Phase 📦 Design Phase 📦 Development Phase 📦 Testing Phase

Components:

- Requirement Concierge

- Business Analyst
- Program Manager
- Solution Architect
- Technical Architect
- API Designer
- Database Architect
- Developer
- Tester
- QA Specialist

Connections:

- RC → BA
- BA → PM
- PM → SA
- SA → TA
- TA → AD
- AD → DBA
- DBA → DEV
- DEV → TEST
- TEST → QA

3. Communication Personas

Specialized personas for managing information flow:

| Persona | Purpose | Key Functions | |---------|---------|---------------| | **Central Knowledge Hub** | Single source of truth | - Store original requirements
- Provide role-appropriate context
- Track interpretation changes | | **Verification Service** | Understanding validation | - Verify persona understanding
- Assess accuracy scores
- Identify clarification needs | | **Collaborative Transition Manager** | Smooth handoffs | - Facilitate collaborative transitions
- Manage knowledge transfer
- Ensure information fidelity |

4. Specialized Domain Personas

Additional expertise for specific domains:

- **Infrastructure Engineer**: DevOps and deployment
- **Security Architect**: Security design and compliance
- **UI/UX Designer**: User experience design
- **Performance Architect**: Scalability and optimization
- **Integration Team Leader**: Cross-team coordination
- **Team Lead Coordinator**: Multi-team management

Persona Intelligence Framework

System Diagram (Top-Bottom Layout)

📦 Input Processing 📦 Persona Core 📦 Output Generation

Components:

- Requirements Input
- Context Analysis
- Historical Learning
- Role Definition
- Domain Expertise
- Pattern Recognition
- Decision Engine
- Response Generation
- Validation
- Format Optimization

Connections:

- REQ → CTX
- CTX → HIS
- HIS → ROLE
- ROLE → EXP
- EXP → PAT
- PAT → DEC
- DEC → RES
- RES → VAL
- VAL → FMT

🔄 Workflow Orchestration

Dynamic Workflow Design Process

Process Flow Sequence

Participants:

- Client
- Orchestrator
- WorkflowDesigner
- TeamArchitect

- CommArchitect
- ExecutionEngine

Process Steps: 1. Client → Orchestrator: Project Requirements 2. Orchestrator → WorkflowDesigner: Design SDLC Workflow 3. WorkflowDesigner- → Orchestrator: Workflow Definition 4. Orchestrator → TeamArchitect: Design Team Structure 5. TeamArchitect- → Orchestrator: Team Structure 6. Orchestrator → CommArchitect: Design Communication Strategy 7. CommArchitect- → Orchestrator: Communication Plan 8. Orchestrator → ExecutionEngine: Execute Designed Workflow 9. ExecutionEngine → ExecutionEngine: Execute Phase 10. ExecutionEngine → ExecutionEngine: Verify Results 11. ExecutionEngine → ExecutionEngine: Transition to Next Phase 12. ExecutionEngine- → Client: Completed Software

Workflow Adaptation Engine

Dynamic Phase Selection:

- AI analyzes project complexity
- Selects appropriate SDLC phases
- Assigns optimal personas to each phase
- Creates dependency mappings

Parallel Processing:

- Identifies parallel work opportunities
- Coordinates multi-team development
- Manages resource allocation
- Prevents workflow bottlenecks

Quality Gates:

- AI-driven quality checkpoints
- Automatic verification processes
- Context fidelity validation
- Requirement alignment assessment

📡 Communication Architecture

Hub-and-Spoke Model

System Diagram (Top-Bottom Layout)

📦 Persona Network 📦 Communication Services

Components:

- Central Knowledge Hub Single Source of Truth
- Requirement Concierge

- Solution Architect
- Developer
- Tester
- Other Personas
- Verification Service
- Collaborative Transition Manager

Connections:

- CKH <-> P1
- CKH <-> P2
- CKH <-> P3
- CKH <-> P4
- CKH <-> P5
- P1 <-> VS
- P2 <-> VS
- P3 <-> VS
- P4 <-> VS
- P1 <-> CTM

Anti-Pattern Prevention

Chinese Whispers Prevention:

- Direct access to original requirements
- Context fidelity tracking
- Understanding verification
- Information loss detection

Communication Quality Assurance:

- Automatic accuracy scoring
- Misunderstanding detection
- Clarification triggers
- Context enrichment

Collaborative Handoffs:

- Joint review sessions
- Knowledge transfer protocols
- Mentorship coordination
- Quality validation

🛠️ Service Infrastructure

Microservices Architecture

System Diagram (Top-Bottom Layout)

📦 API Gateway Layer 📦 Core Services 📦 AI Services 📦 Data Services 📦 Infrastructure

Components:

- API Gateway Load Balancer
- Personas Gateway Port 8013
- RAG Engine Port 8003
- Secrets Management Port 8004
- Port Management Port 8005
- AI Manager Port 8007
- Pinecone Index Port 8006
- (PostgreSQL Port 5432)
- (Redis Port 6379)
- Monitoring Prometheus/Grafana
- Logging ELK Stack

Connections:

- GW → PG
- GW → RAG
- PG → RAG
- PG → SM
- RAG → AI
- RAG → PIN
- AI → PIN
- PG → PDB
- RAG → PDB
- PG → RDB

Service Specifications

Service	Port	Technology	Purpose	Scaling
Personas Gateway	8013	FastAPI + Python	Persona interface management	Horizontal
RAG Engine	8003	Python + LangChain	AI processing engine	Horizontal
Secrets Management	8004	FastAPI + Vault	Secure credential management	Redundant
Port Management	8005	FastAPI + SQLite	Dynamic port allocation	Single instance
AI Manager	8007	Python + AI SDKs	AI model orchestration	Horizontal
Pinecone Index	8006	Python + Pinecone	Vector database interface	Horizontal

Container Orchestration

Docker Compose Architecture

```
version: '3.8'

services:
  personas-gateway:
    image: g1/personas-gateway:latest
    ports: ["8013:8013"]
    depends_on: [rag-engine, secrets-management]
    environment:
      - RAG_ENGINE_URL=http://rag-engine:8003
      - SECRETS_SERVICE_URL=http://secrets-management:8004
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8013/health"]
      interval: 30s
      timeout: 10s
      retries: 3



   rag-engine:
    image: g1/rag-engine:latest
    ports: ["8003:8003"]
    depends_on: [secrets-management, postgres, redis]
    environment:
      - POSTGRES_URL=postgresql://postgres:5432/rag_db
      - REDIS_URL=redis://redis:6379
      - SECRETS_SERVICE_URL=http://secrets-management:8004
```

🔄 Data Flow & Integration

Request Processing Flow

Process Flow Sequence

Participants:

- Client
- Gateway
- Orchestrator
- KnowledgeHub
- Personas
- RAGEngine
- Database

Process Steps: 1. Client → Gateway: Project Request 2. Gateway → Orchestrator: Validated Request 3. Orchestrator → KnowledgeHub: Store Original Requirements 4. KnowledgeHub → Database: Persist Context 5. Orchestrator → KnowledgeHub: Get Context for Persona 6. KnowledgeHub → Personas: Role-appropriate Context 7. Personas → RAGEngine: Process with AI 8. RAGEngine → Database: Query Relevant Information 9. RAGEngine- → Personas: AI-generated Response 10. Personas → KnowledgeHub: Update with Results 11. KnowledgeHub → Database: Store Results 12. Orchestrator- → Client: Final Software Product

Context Management

Context Types:

- **Original Requirements**: Immutable source of truth
- **Role Context**: Persona-specific information
- **Workflow Context**: Phase and dependency information
- **Historical Context**: Learning from previous executions

Context Enrichment:

- Automatic context expansion
- Related information retrieval
- Cross-reference validation
- Quality scoring

Context Preservation:

- Immutable requirement storage
- Version-controlled context
- Audit trail maintenance
- Rollback capabilities

🔐 Security & Compliance

Security Architecture

System Diagram (Top-Bottom Layout)

📦 **External Layer** 📦 **Authentication Layer** 📦 **Application Layer** 📦 **Infrastructure Layer**

Components:

- Web Application Firewall
- Load Balancer SSL Termination
- JWT Authentication
- Role-Based Access Control
- API Key Management

- Data Encryption at Rest & Transit
- Input Validation
- Rate Limiting
- Network Isolation VPC/Subnets
- Security Monitoring SIEM
- Audit Logging

Connections:

- WAF → LB
- LB → AUTH
- AUTH → RBAC
- RBAC → API
- API → ENC
- ENC → VAL
- VAL → RATE
- RATE → NET
- NET → MON
- MON → AUDIT

Security Features

Authentication & Authorization:

- JWT-based authentication
- Role-based access control (RBAC)
- API key management
- Service-to-service authentication

Data Protection:

- Encryption at rest (AES-256)
- Encryption in transit (TLS 1.3)
- Secure secret management
- Data anonymization capabilities

Compliance Standards:

- **GDPR**: Data privacy and user rights
- **SOC 2**: Security and availability controls
- **ISO 27001**: Information security management
- **HIPAA**: Healthcare data protection (when applicable)

Security Monitoring:

- Real-time threat detection
- Anomaly detection

- Security audit logging
- Compliance reporting

📊 Scalability & Performance

Horizontal Scaling Architecture

System Diagram (Top-Bottom Layout)

📦 Load Balancer 📦 Auto Scaling Groups 📦 Personas Gateway Cluster 📦 RAG Engine Cluster 📦 Database Cluster

Components:

- Application Load Balancer
- Personas Gateway 1
- Personas Gateway 2
- Personas Gateway N
- RAG Engine 1
- RAG Engine 2
- RAG Engine N
- (PostgreSQL Primary)
- (PostgreSQL Replica 1)
- (PostgreSQL Replica N)
- (Redis Cluster)

Connections:

- ALB → PG1
- ALB → PG2
- ALB → PG3
- PG1 → RAG1
- PG2 → RAG2
- PG3 → RAG3
- RAG1 → PDB_PRIMARY
- RAG2 → PDB_REPLICA1
- RAG3 → PDB_REPLICA2
- PG1 → RDB_CLUSTER

Performance Optimization

Caching Strategy:

- **L1 Cache**: In-memory response caching
- **L2 Cache**: Redis distributed caching
- **L3 Cache**: CDN for static content
- **Smart Invalidation**: Context-aware cache management

Async Processing:

- Non-blocking I/O operations
- Concurrent persona processing
- Background task queuing
- Streaming responses

Database Optimization:

- Connection pooling
- Query optimization
- Index management
- Read replicas

Performance Metrics:

- **Response Time**: <200ms average
- **Throughput**: 10,000+ requests/second
- **Availability**: 99.9% uptime SLA
- **Scalability**: Auto-scaling based on demand

🚀 Deployment Architecture

Cloud-Native Deployment

System Diagram (Top-Bottom Layout)

📦 Production Environment 📦 Kubernetes Cluster 📦 Ingress 📦 Application Pods 📦 Storage 📦 Managed Services 📦 Monitoring

Components:

- NGINX Ingress Controller
- Cert Manager
- Personas Gateway Pods
- RAG Engine Pods
- AI Manager Pods
- Persistent Volume Claims
- Kubernetes Secrets

- (AWS RDS PostgreSQL)
- (AWS ElastiCache)
- (AWS S3 Storage)
- Prometheus
- Grafana
- Jaeger Tracing

Connections:

- ING → PG_POD
- ING → RAG_POD
- PG_POD → RAG_POD
- PG_POD → AI_POD
- PG_POD → RDS
- RAG_POD → RDS
- PG_POD → ELASTICACHE
- RAG_POD → ELASTICACHE
- PG_POD → S3
- RAG_POD → S3

Environment Management

Multi-Environment Strategy:

- **Development**: Local Docker Compose
- **Staging**: Kubernetes cluster with production-like setup
- **Production**: Multi-region Kubernetes deployment
- **Disaster Recovery**: Cross-region backup and failover

CI/CD Pipeline:

System Diagram (Left-Right Layout)

Connections:

- DEV[Developer] → GIT[Git Repository]
- GIT → BUILD[Build & Test]
- BUILD → SCAN[Security Scan]
- SCAN → STAGE[Deploy to Staging]
- STAGE → TEST[Integration Tests]
- TEST → APPROVE[Manual Approval]
- APPROVE → PROD[Deploy to Production]
- PROD → MONITOR[Monitor & Validate]

Deployment Configuration:

- **Blue-Green Deployments**: Zero-downtime updates

- **Canary Releases**: Gradual feature rollouts
- **Feature Flags**: Runtime feature control
- **Health Checks**: Automated health validation

📈 Monitoring & Observability

Observability Stack

System Diagram (Top-Bottom Layout)

📦 Data Collection 📦 Processing Layer 📦 Visualization 📦 Alerting

Components:

- Application Metrics
- Application Logs
- Distributed Traces
- Prometheus Metrics Storage
- ELK Stack Log Processing
- Jaeger Trace Processing
- Grafana Dashboards
- Kibana Log Analysis
- Jaeger UI Trace Analysis
- Alert Manager
- PagerDuty
- Slack Notifications

Connections:

- APPS → PROM
- LOGS → ELK
- TRACES → JAEGER
- PROM → GRAF
- ELK → KIB
- JAEGER → JAEGER_UI
- PROM → AM
- AM → PD
- AM → SLACK

Key Metrics & KPIs

Business Metrics:

- Project completion rate
- Average development time
- Customer satisfaction score
- Cost per project

Technical Metrics:

- API response times
- Error rates
- Throughput (requests/second)
- Resource utilization

AI Performance Metrics:

- Persona response accuracy
- Context preservation rate
- Workflow adaptation success
- Requirement fidelity score

🎯 Technology Stack Summary

Backend Technologies

- **Languages**: Python 3.11+, JavaScript/TypeScript
- **Frameworks**: FastAPI, AsyncIO, LangChain
- **Databases**: PostgreSQL, Redis, Vector Databases
- **AI/ML**: OpenAI API, Claude API, Hugging Face
- **Message Queue**: Redis Pub/Sub, Celery

Infrastructure Technologies

- **Containerization**: Docker, Docker Compose
- **Orchestration**: Kubernetes
- **Cloud Platforms**: AWS, GCP, Azure
- **Service Mesh**: Istio (optional)
- **API Gateway**: NGINX, Kong

Development & Operations

- **Version Control**: Git, GitHub
- **CI/CD**: GitHub Actions, Jenkins
- **Monitoring**: Prometheus, Grafana, ELK Stack

- **Security**: HashiCorp Vault, JWT, OAuth2
- **Testing**: pytest, Jest, Kubernetes test frameworks

🚀 Future Architecture Evolution

Planned Enhancements

Advanced AI Integration:

- Multi-modal AI capabilities (text, image, video)
- Domain-specific AI model fine-tuning
- Federated learning across deployments
- Advanced reasoning capabilities

Enhanced Scalability:

- Multi-cloud deployment
- Edge computing integration
- Global content delivery network
- Advanced auto-scaling algorithms

Extended Capabilities:

- Real-time collaboration features
- Voice-to-code generation
- Automated testing and deployment
- Intelligent project management

Industry-Specific Modules:

- Healthcare compliance module
- Financial services security
- Manufacturing integration
- Education-specific features

📞 Architecture Support & Documentation

Documentation Resources

- **API Documentation**: OpenAPI/Swagger specifications

- **Deployment Guides**: Step-by-step deployment instructions
- **Development Setup**: Local development environment guide
- **Troubleshooting**: Common issues and solutions

Support Channels

- **Technical Documentation**: Comprehensive architecture guides
- **Developer Community**: Forums and discussion boards
- **Professional Support**: Enterprise support options
- **Training Resources**: Architecture and integration training

---

Document Status: ✅ Complete **Last Updated**: 2025-08-27 **Version Control**: Maintained in Git repository **Review Cycle**: Quarterly architecture reviews

---

This architecture document represents the current state of the G1 platform and serves as the definitive guide for understanding, deploying, and extending the system.