The 50-Point Software Quality Assessment Checklist for CTOs

A Comprehensive Self-Assessment Tool for Evaluating Your Software Development Organization

Techsessment Consulting LLC | www.techsessment.com

How to Use This Checklist

This assessment covers five critical dimensions of software quality: Architecture & Design, Code Quality & Maintainability, Testing & Quality Assurance, Security & Compliance, and Operations & DevOps.

3 points	Fully implemented with documented processes and metrics
2 points	Partially implemented with room for improvement
1 point	Minimal implementation or inconsistent application
0 points	Not implemented or significant gaps exist

Interpretation:

120–150 points	Excellent — world-class software quality practices
90–119 points	Good — solid foundation with targeted improvement opportunities
60–89 points	Fair — significant gaps requiring systematic improvement
Below 60	Critical — immediate attention needed to reduce risk

Section 1: Architecture & Design (30 Points)

System Architecture

1. Architectural Documentation

■ **3** — Comprehensive architecture documentation with diagrams (C4 model or equivalent), decision records, and regular updates

■ **2** — Basic architecture diagrams exist but may be outdated or incomplete

■ **1** — Minimal documentation, mostly in developers' heads

■ **0** — No architectural documentation exists

Score: _____

2. Separation of Concerns

- ■ **3** — Clear layered architecture with well-defined boundaries (presentation, business logic, data access)
- ■ **2** — Layers exist but boundaries are occasionally violated
- ■ **1** — Minimal separation, significant mixing of concerns
- ■ **0** — Monolithic design with no separation

Score: _____

3. Dependency Management

- ■ **3** — Dependencies point inward following Dependency Inversion Principle, no circular dependencies
- ■ **2** — Generally good dependency structure with some circular references
- ■ **1** — Frequent circular dependencies creating tight coupling
- ■ **0** — Chaotic dependency graph with no clear structure

Score: _____

4. Scalability Design

- ■ **3** — Architecture designed for horizontal scaling with documented capacity planning
- ■ **2** — Some scalability considerations, limited to specific components
- ■ **1** — Minimal scalability design, primarily vertical scaling
- ■ **0** — No scalability considerations in architecture

Score: _____

5. API Design & Standards

- ■ **3** — Consistent API design patterns (REST/GraphQL), versioning strategy, comprehensive documentation
- ■ **2** — APIs exist with basic documentation but inconsistent patterns
- ■ **1** — Ad-hoc API design with minimal documentation
- ■ **0** — No standardized API approach

Score: _____

Design Principles

6. SOLID Principles Adherence

- ■ **3** — Code consistently follows SOLID principles with documented examples and team training
- ■ **2** — Team understands SOLID, applied in most new code
- ■ **1** — Limited application of SOLID principles
- ■ **0** — No awareness or application of SOLID principles

Score: _____

7. Design Pattern Usage

- ■ **3** — Appropriate design patterns used consistently with team knowledge sharing
- ■ **2** — Some patterns used, not consistently applied across codebase
- ■ **1** — Minimal pattern usage, mostly ad-hoc solutions
- ■ **0** — No design pattern awareness or application

Score: _____

8. Database Design Quality

- ■ **3** — Normalized database design (appropriate level), proper indexing, documented schema with ER diagrams
- ■ **2** — Generally good design with some optimization opportunities
- ■ **1** — Functional but poorly optimized database design
- ■ **0** — Significant database design issues impacting performance

Score: _____

9. Service Boundaries (Microservices/Modules)

- ■ **3** — Well-defined service boundaries based on bounded contexts, minimal cross-service dependencies
- ■ **2** — Services defined but some boundary violations exist
- ■ **1** — Services exist but boundaries are poorly defined
- ■ **0** — Monolithic architecture or poorly separated services

Score: _____

10. Technical Debt Management

- ■ **3** — Technical debt tracked in backlog, regular debt reduction sprints, documented architecture decision records
- ■ **2** — Technical debt identified but limited systematic reduction
- ■ **1** — Technical debt acknowledged but not actively managed
- ■ **0** — Technical debt accumulating with no tracking or management

Score: _____

Section 1 Total: _____ / 30

Section 2: Code Quality & Maintainability (30 Points)

Code Standards

11. Coding Style Consistency

- ■ **3** — Automated linting enforced in CI/CD, documented style guide, consistent across entire codebase
- ■ **2** — Style guide exists, partially enforced
- ■ **1** — Informal standards, inconsistently applied
- ■ **0** — No coding standards, highly inconsistent code

Score: _____

12. Code Readability

- ■ **3** — Code is self-documenting with clear naming, appropriate comments explaining 'why' not 'what'
- ■ **2** — Generally readable with some areas needing improvement
- ■ **1** — Code difficult to understand without significant effort
- ■ **0** — Code is cryptic, relies heavily on original author knowledge

Score: _____

13. Code Complexity Management

- ■ **3** — Cyclomatic complexity monitored, functions/methods under 10-15 complexity, no 'god objects'
- ■ **2** — Complexity generally managed but some hot spots exist
- ■ **1** — Frequent high-complexity methods and classes
- ■ **0** — Extreme complexity making code unmaintainable

Score: _____

14. DRY Principle (Don't Repeat Yourself)

- ■ **3** — Minimal code duplication, well-factored shared components and utilities
- ■ **2** — Some duplication exists but major logic is shared
- ■ **1** — Significant code duplication across modules
- ■ **0** — Extensive copy-paste programming

Score: _____

15. Code Documentation

- ■ **3** — Comprehensive inline documentation, auto-generated API docs, architecture decision records
- ■ **2** — Basic documentation exists but could be more comprehensive
- ■ **1** — Minimal documentation, mostly in README files
- ■ **0** — No meaningful documentation

Score: _____

Code Review Practices

16. Code Review Process

- ■ **3** — Mandatory code reviews for all changes, documented checklist, timely reviews (< 24 hours)
- ■ **2** — Code reviews happen but not consistently enforced
- ■ **1** — Occasional code reviews, no formal process
- ■ **0** — No code review process

Score: _____

17. Pull Request Quality

- ■ **3** — Small, focused PRs (< 500 lines), clear descriptions, automated checks pass before review
- ■ **2** — PRs generally well-structured but occasionally too large
- ■ **1** — Large, unfocused PRs common
- ■ **0** — No PR standards or process

Score: _____

18. Code Review Culture

- ■ **3** — Constructive feedback culture, reviews focus on learning, team sees reviews as valuable
- ■ **2** — Reviews happen but can be perfunctory or delayed
- ■ **1** — Reviews seen as bottleneck or rubber stamp
- ■ **0** — No review culture established

Score: _____

Refactoring & Maintenance

19. Refactoring Practices

- ■ **3** — Regular refactoring cycles, boy scout rule applied, dedicated refactoring time allocated
- ■ **2** — Opportunistic refactoring when touching code
- ■ **1** — Minimal refactoring, only when absolutely necessary
- ■ **0** — No refactoring, fear of breaking existing code

Score: _____

20. Legacy Code Management

- ■ **3** — Strategy for modernizing legacy code, characterization tests in place, incremental improvement plan
- ■ **2** — Some legacy code improvement efforts underway
- ■ **1** — Legacy code acknowledged but no systematic approach
- ■ **0** — Legacy code growing, no management strategy

Score: _____

Section 2 Total: _____ / 30

Section 3: Testing & Quality Assurance (30 Points)

Test Coverage & Strategy

21. Unit Test Coverage

- ■ **3** — 80%+ coverage for business logic, tests run on every commit, coverage tracked over time
- ■ **2** — 50-80% coverage, gaps in some critical areas
- ■ **1** — < 50% coverage, inconsistent testing
- ■ **0** — Minimal or no unit tests

Score: _____

22. Test Quality

- ■ **3** — Tests are fast (< 10 min suite), reliable (no flaky tests), readable, follow AAA pattern
- ■ **2** — Tests exist but may be slow or occasionally flaky
- ■ **1** — Tests unreliable or difficult to maintain
- ■ **0** — Tests non-existent or never run

Score: _____

23. Integration Testing

- ■ **3** — Comprehensive integration tests for all major workflows, automated and part of CI/CD
- ■ **2** — Some integration tests exist for critical paths
- ■ **1** — Minimal integration testing
- ■ **0** — No integration tests

Score: _____

24. End-to-End Testing

- ■ **3** — Automated E2E tests for critical user journeys, run regularly, stable test infrastructure
- ■ **2** — Some E2E tests but may be manual or unreliable
- ■ **1** — Primarily manual testing with no automation
- ■ **0** — No systematic E2E testing

Score: _____

25. Test-Driven Development (TDD)

- ■ **3** — TDD practiced for new features, team trained in TDD, measurable quality improvements
- ■ **2** — TDD used occasionally for complex logic
- ■ **1** — Tests written after code consistently
- ■ **0** — No TDD practices

Score: _____

Quality Assurance Process

26. Continuous Integration

- ■ **3** — All tests run on every commit, builds fast (< 10 min), clear failure notifications
- ■ **2** — CI configured but builds may be slow or unstable
- ■ **1** — Basic CI setup with frequent issues
- ■ **0** — No CI pipeline

Score: _____

27. Static Code Analysis

- ■ **3** — Automated static analysis (SonarQube, CodeClimate), quality gates enforced, trends monitored
- ■ **2** — Static analysis runs but not consistently enforced
- ■ **1** — Occasional manual static analysis
- ■ **0** — No static code analysis

Score: _____

28. Performance Testing

- ■ **3** — Regular performance testing, benchmarked against SLAs, automated performance regression detection
- ■ **2** — Occasional performance testing for major features
- ■ **1** — Performance testing only when problems occur
- ■ **0** — No performance testing

Score: _____

29. Bug Tracking & Management

- ■ **3** — All bugs tracked in system, prioritized by severity, metrics on resolution time and patterns
- ■ **2** — Bugs tracked but prioritization could improve
- ■ **1** — Informal bug tracking in scattered locations
- ■ **0** — No systematic bug tracking

Score: _____

30. Regression Testing

- ■ **3** — Automated regression suite covers all major features, runs before every release
- ■ **2** — Some automated regression tests exist
- ■ **1** — Primarily manual regression testing
- ■ **0** — No regression testing strategy

Score: _____

Section 3 Total: _____ / 30

Section 4: Security & Compliance (30 Points)

Security Practices

31. Security Code Review

- ■ **3** — Dedicated security review for all changes touching sensitive areas, security checklist used
- ■ **2** — Security considerations in general code reviews
- ■ **1** — Ad-hoc security reviews
- ■ **0** — No security-focused reviews

Score: _____

32. Authentication & Authorization

- ■ **3** — Industry-standard authentication (OAuth 2.0/OIDC), role-based access control, regular audits
- ■ **2** — Basic authentication/authorization implemented
- ■ **1** — Minimal security controls
- ■ **0** — Weak or no authentication/authorization

Score: _____

33. Input Validation & Sanitization

- ■ **3** — All user input validated and sanitized, protection against injection attacks, automated testing
- ■ **2** — Input validation in most places but gaps exist
- ■ **1** — Inconsistent input validation
- ■ **0** — Little to no input validation

Score: _____

34. Secrets Management

- ■ **3** — All secrets in secure vault (HashiCorp Vault, AWS Secrets Manager), no secrets in code/repos
- ■ **2** — Environment variables used but some secrets might be in config files
- ■ **1** — Many secrets in configuration files
- ■ **0** — Secrets hardcoded in source code

Score: _____

35. Dependency Vulnerability Scanning

- ■ **3** — Automated scanning (Snyk, Dependabot), vulnerabilities addressed within SLA, regular updates
- ■ **2** — Occasional dependency scans and updates
- ■ **1** — Manual dependency checking, infrequent updates
- ■ **0** — No dependency vulnerability management

Score: _____

36. Data Encryption

- ■ **3** — Data encrypted at rest and in transit (TLS 1.2+), encryption key management, regular reviews
- ■ **2** — Basic encryption for sensitive data
- ■ **1** — Minimal encryption coverage
- ■ **0** — No systematic encryption strategy

Score: _____

37. Security Testing

- ■ **3** — Regular penetration testing, automated security scans in CI/CD, OWASP Top 10 testing
- ■ **2** — Occasional security testing
- ■ **1** — Security testing only before major releases
- ■ **0** — No security testing

Score: _____

38. Security Incident Response

- ■ **3** — Documented incident response plan, regular drills, clear escalation paths
- ■ **2** — Basic incident response procedures exist
- ■ **1** — Ad-hoc response to security incidents
- ■ **0** — No incident response planning

Score: _____

Compliance & Governance

39. Data Privacy Compliance

- ■ **3** — GDPR/CCPA compliant, documented data handling procedures, privacy by design
- ■ **2** — Basic privacy controls, working toward compliance
- ■ **1** — Minimal privacy considerations
- ■ **0** — No data privacy program

Score: _____

40. Audit Logging

- ■ **3** — Comprehensive audit logs for all sensitive operations, tamper-proof, regular review
- ■ **2** — Basic logging for major events
- ■ **1** — Minimal logging infrastructure
- ■ **0** — No audit logging

Score: _____

Section 4 Total: _____ / 30

Section 5: Operations & DevOps (30 Points)

Deployment & Release

41. Deployment Automation

- ■ **3** — Fully automated deployments, zero-downtime deployments, rollback capability
- ■ **2** — Automated deployments with some manual steps
- ■ **1** — Primarily manual deployments with scripts
- ■ **0** — Fully manual deployment process

Score: _____

42. Release Process

- ■ **3** — Documented release process, release notes automated, regular release cadence
- ■ **2** — Basic release process with some documentation
- ■ **1** — Ad-hoc releases with minimal process
- ■ **0** — No defined release process

Score: _____

43. Environment Management

- ■ **3** — Dev/staging/production parity, infrastructure as code, environments easily reproducible
- ■ **2** — Multiple environments but some configuration drift
- ■ **1** — Basic environment separation
- ■ **0** — No environment separation or management

Score: _____

44. Configuration Management

- ■ **3** — All configuration in version control, environment-specific configs, no manual server configuration
- ■ **2** — Most configuration managed, some manual steps
- ■ **1** — Minimal configuration management
- ■ **0** — Ad-hoc configuration changes

Score: _____

Monitoring & Observability

45. Application Monitoring

- ■ **3** — Comprehensive monitoring (APM, logs, metrics), alerts configured, dashboards for key metrics
- ■ **2** — Basic monitoring with limited visibility
- ■ **1** — Minimal monitoring, primarily reactive
- ■ **0** — No application monitoring

Score: _____

46. Error Tracking & Alerting

- ■ **3** — Automated error tracking (Sentry, Rollbar), alerts route to on-call, error budgets tracked
- ■ **2** — Error logging exists but limited alerting
- ■ **1** — Manual log checking for errors
- ■ **0** — No systematic error tracking

Score: _____

47. Performance Monitoring

- ■ **3** — Real-time performance metrics, SLA tracking, proactive performance optimization
- ■ **2** — Basic performance metrics collected
- ■ **1** — Occasional performance checks
- ■ **0** — No performance monitoring

Score: _____

48. Incident Management

- ■ **3** — Documented incident response, post-mortems conducted, on-call rotation, runbooks
- ■ **2** — Basic incident response procedures
- ■ **1** — Ad-hoc incident handling
- ■ **0** — No incident management process

Score: _____

Documentation & Knowledge Transfer

49. Operational Documentation

- ■ **3** — Comprehensive runbooks, architecture diagrams current, disaster recovery documented
- ■ **2** — Basic operational documentation exists
- ■ **1** — Minimal documentation, mostly tribal knowledge
- ■ **0** — No operational documentation

Score: _____

50. Knowledge Sharing Culture

- ■ **3** — Regular tech talks, documentation time allocated, pair programming practiced, new hire onboarding process
- ■ **2** — Some knowledge sharing activities
- ■ **1** — Minimal knowledge sharing
- ■ **0** — Knowledge siloed with individuals

Score: _____

Section 5 Total: _____ / 30

Assessment Summary

Section	Your Score	Maximum	Percentage
Architecture & Design	_____	30	_____%
Code Quality & Maintainability	_____	30	_____%
Testing & Quality Assurance	_____	30	_____%
Security & Compliance	_____	30	_____%
Operations & DevOps	_____	30	_____%
TOTAL	_____	**150**	_____%%

Interpreting Your Results

120–150 points (Excellent)

Your organization demonstrates world-class software quality practices. Focus on maintaining standards, continuous improvement, and sharing best practices industry-wide. Consider minor optimizations in areas scoring below 2.5.

90–119 points (Good)

You have a solid foundation with established processes and practices. Prioritize the 2-3 lowest-scoring sections for systematic improvement. Consider Techsessment's targeted assessment services to accelerate progress in specific areas.

60–89 points (Fair)

Significant quality gaps exist that likely impact velocity, reliability, and team morale. A comprehensive quality assessment is recommended to create a systematic improvement roadmap. Focus on quick wins in sections scoring below 2.0 while building long-term improvement plans.

Below 60 points (Critical)

Immediate attention is needed. Quality issues are likely causing production incidents, customer dissatisfaction, and developer frustration. Consider Techsessment's Complete Package for a systematic approach to building quality practices from the foundation up.

Section-Specific Guidance

Architecture & Design (Low Score)

- **Immediate Action:** Document current architecture state, identify critical circular dependencies
- **Short-term (1-3 months):** Establish architecture decision record process, create basic architecture diagrams
- **Long-term (3-12 months):** Implement systematic debt reduction, establish architectural review board

Code Quality & Maintainability (Low Score)

- **Immediate Action:** Implement automated linting, establish code review checklist
- **Short-term:** Document coding standards, conduct team training on SOLID principles
- **Long-term:** Reduce code complexity systematically, establish refactoring cycles

Testing & Quality Assurance (Low Score)

- **Immediate Action:** Set up basic CI pipeline, establish coverage baseline
- **Short-term:** Implement unit testing for new code, create integration test framework
- **Long-term:** Achieve 80%+ coverage, implement TDD practices, establish automated E2E testing

Security & Compliance (Low Score)

- **Immediate Action:** Move secrets to environment variables, enable dependency scanning
- **Short-term:** Implement input validation framework, establish security review checklist
- **Long-term:** Achieve compliance certification, implement comprehensive security testing program

Operations & DevOps (Low Score)

- **Immediate Action:** Document current deployment process, set up basic error logging
- **Short-term:** Automate deployments, implement monitoring for critical paths
- **Long-term:** Achieve full infrastructure as code, implement comprehensive observability

Next Steps: How Techsessment Can Help

Based on your assessment results, Techsessment Consulting LLC offers targeted services to address your specific quality challenges, delivered through a proven sequential methodology that ensures each service builds on the last.

If You Scored 90–119 (Good) — Targeted Gaps

Recommended: Essential Package + Add-On Services

- Begins with Architecture Review, Technical Debt Assessment, and Tech Stack Evaluation (our Foundation tier)
- Additional services added sequentially based on your lowest-scoring sections
- Investment: $15,000 (Essential Package) + $5,000 per additional service

If You Scored 60–89 (Fair) — Systematic Improvement

Recommended: Complete Package

- All 10 services delivered in our proven four-tier sequence: Foundation → Analysis → Process → Strategic
- Comprehensive improvement roadmap addressing every dimension of your assessment
- Investment: $45,000 (all 10 services over 10 weeks)

If You Scored Below 60 (Critical) — Urgent Foundation Work

Recommended: Complete Package — Priority Engagement

- Full sequential delivery starting immediately with Foundation services
- Every service prerequisite is honored to ensure sustainable, lasting improvement
- Investment: $45,000 — the fastest path to a stable, quality engineering organization

Schedule Your Free 30-Minute Discovery Call

Let's discuss your assessment results and determine the best path forward for your organization. No sales pressure — just an honest conversation about your software quality and whether we can add value.

contact@techsessment.com | www.techsessment.com

About Techsessment Consulting LLC

Techsessment Consulting LLC brings deep software quality expertise to growth-stage technology companies across Texas and nationwide. Our systematic, documentation-first approach delivers comprehensive assessments your team can act on immediately — without the overhead of training, coaching, or ongoing retainers.

- **Sequential Methodology:** Services are delivered in a proven four-tier sequence — Foundation, Analysis, Process, Strategic — achieving 85% implementation success versus 35% with cherry-picked approaches
- **Actionable Deliverables:** Comprehensive Markdown documentation your team can implement at their own pace
- **Developer Perspective:** Founded by a developer, not a marketing guy — we speak your language and understand your real challenges
- **Texas-Based, Nationwide Reach:** Local roots with the expertise to serve companies across the country

Our 10-Service Framework (delivered sequentially):

Foundation Tier	Analysis Tier	Process Tier	Strategic Tier
Architecture Review	Technology Stack Evaluation	Code Review Standards	Feature Review & Planning
Technical Debt Assessment	Performance Optimization	Documentation & Knowledge Transfer	Developer Onboarding
	Refactoring Roadmap	Unit Testing Strategy	

© 2025 Techsessment Consulting LLC. All rights reserved. This checklist is provided as a free resource for CTOs and engineering leaders. You are welcome to share it within your organization. For external distribution, please maintain attribution to Techsessment Consulting LLC.