GUIDE

Software Risk Assessment Checklist

> A software risk assessment scores a program against known predictors of failure — requirements quality, schedule realism, process discipline, architectural complexity, test coverage and edge case handling — and converts the weakest areas into specific, owned mitigations rather than a generic risk register.

How to use this checklist

Score each item as green, yellow or red with evidence, not opinion. Any red on a safety- or mission-critical function becomes a tracked risk with an owner, a mitigation and a due date. Re-score at each major milestone; the trend matters more than the absolute score.

1. Requirements

- Are error handling and off-nominal behaviors specified, not just nominal behavior?
- Are there negative requirements — what the system must not do?
- Is every requirement testable as written?
- Are units, ranges, precision and timing stated for every interface value?
- Is requirements volatility being measured, and is it decreasing?

2. Schedule and resources

- Does the planned test time allow the predicted number of defects to actually be found?
- Has the schedule compressed while scope stayed constant?

- Is integration scheduled late with no slack behind it?
- Is key domain knowledge concentrated in one or two people?

3. Process discipline

- Are peer reviews performed on requirements and design, not only on code?
- Is static analysis run and are its findings actually dispositioned?
- Is bidirectional traceability maintained from requirements to tests?
- Are defects recorded with enough detail — trigger, state, inputs — to support root cause analysis?

4. Architecture and complexity

- How many external interfaces and third-party dependencies exist, and are their failure behaviors defined?
- Is there a defined degraded mode and safe state?
- Is startup, shutdown and restart behavior specified from every state?
- Are concurrency and timing constraints documented and analyzed?
- Is reused or legacy code carrying assumptions nobody currently owns?

5. Test coverage and edge cases

- Do tests exercise invalid, boundary, out-of-order and duplicated inputs — not just valid ones?
- Are resource exhaustion and long-duration conditions tested?
- Are dependency failures injected, or only assumed to work?
- Do the most common edge case categories each map to at least one test?

6. Reliability engineering coverage

- Is there a software reliability program plan?
- Has a prediction been performed and refreshed?
- Has a software FMEA or fault tree analysis been performed on critical functions?
- Is software represented in the system reliability model?
- Do software failures flow through FRACAS?

Turning the checklist into action

Finding	Typical mitigation
Off-nominal behavior unspecified	Add derived requirements from failure mode analysis
Test time insufficient for predicted defects	Rescope, add test capacity, or accept and document the residual risk
No traceability	Establish requirement-to-test mapping for critical functions first
Undefined degraded mode	Define safe state and specify transition conditions
Edge cases untested	Generate tests from an edge case checklist, not from imagination

Frequently asked questions

▼ What are the strongest early warning signs of software program risk?

Unspecified off-nominal behavior, high requirements volatility late in the program, a test schedule that cannot absorb the predicted defect count, and integration scheduled with no slack behind it.

▼ How often should a software risk assessment be repeated?

At every major milestone, and any time scope, schedule or team composition changes materially. The trend between assessments is the useful signal.

▼ Who should perform the assessment?

Someone independent of the delivery team, using evidence from artifacts and defect data rather than status reports.

Related pages

Reliable software risk assessment Identify software program risks training Most common edge cases Software FMEA vs hardware FMEA