GUIDE

Software Reliability Prediction Tool: What It Does and How It Works

> A software reliability prediction tool estimates how many defects a release will contain, how often it will fail and how available it will be — before the code is written — by analyzing requirements, development practices and historical defect data instead of waiting for test results.

Why predict instead of measure?

Traditional software reliability measurement starts when testing starts. By then the architecture is fixed, the requirements are frozen and the cost of change is highest. Prediction moves the analysis to the requirements and design phase, where changes are still cheap, so reliability becomes a design input rather than a test-phase discovery.

Hardware engineers have predicted failure rates from parts counts and stress models for decades. Software reliability prediction applies the same logic: known characteristics of the product and the process are mapped to expected defect behavior using models published in IEEE 1633, the IEEE Recommended Practice on Software Reliability.

What a prediction tool actually analyzes

- **Requirements quality** — ambiguity, missing negative requirements, undefined error handling and untestable statements are leading indicators of downstream defects.
- **Development and test practices** — the presence or absence of practices such as peer reviews, static analysis, requirements traceability and fault-injection testing shifts expected defect density.

- **Size and change history** — new, modified and reused code carry very different defect expectations.
- **Historical defect data** — defects observed on comparable programs calibrate the prediction to your domain.
- **Edge case and failure mode coverage** — the defects that escape to the field are usually the conditions nobody wrote a requirement for.

What the outputs look like

Output	How it is used
Predicted defect count and defect density	Size the test effort and the defect backlog before test starts
Predicted failure rate / MTBF	Feed software into the system reliability model alongside hardware
Predicted availability	Support error budgets, SLAs and mission availability requirements
Defect discovery profile over time	Decide whether the schedule allows the defects to actually be found
Ranked risks and missing practices	Target the specific practices that move the prediction most

Prediction vs. reliability growth models

Prediction and growth modeling answer different questions and are used at different times.

	Prediction	Reliability growth model
When	Requirements and design	During and after test
Input	Requirements, process, historical data	Observed failure data
Question answered	How many defects will we have?	How many remain, and when can we ship?

Mature programs use both: prediction to plan, growth models to confirm.

How to evaluate a prediction tool

1. Does it work from requirements, before code exists — or does it need failure data?
2. Is the model traceable to a published standard such as IEEE 1633 or SAE J1025?
3. Does it produce a failure rate that can enter the system reliability model, not just a defect count?
4. Does it tell you which practices to change, not just what the number is?
5. Can it be re-run as the requirements evolve?

Frequently asked questions

▼ **Can software defects really be predicted before coding starts?**

Yes. Defect density correlates strongly with requirements quality, development practices, code size and reuse. Those inputs exist before code does, which is why IEEE 1633 defines prediction models for exactly this phase.

▼ How accurate is software defect prediction?

Accuracy depends on how well the model is calibrated to your domain and how honestly the process inputs are answered. Predictions are used as planning ranges and relative comparisons between options, not as single exact numbers.

▼ Does prediction replace testing?

No. It sizes and targets testing. Prediction tells you roughly how many defects to expect and where they concentrate, so test effort can be planned rather than discovered.

Related pages

All Requs AI products Predict Software Defects training IEEE 1633 IEEE 1633 explained