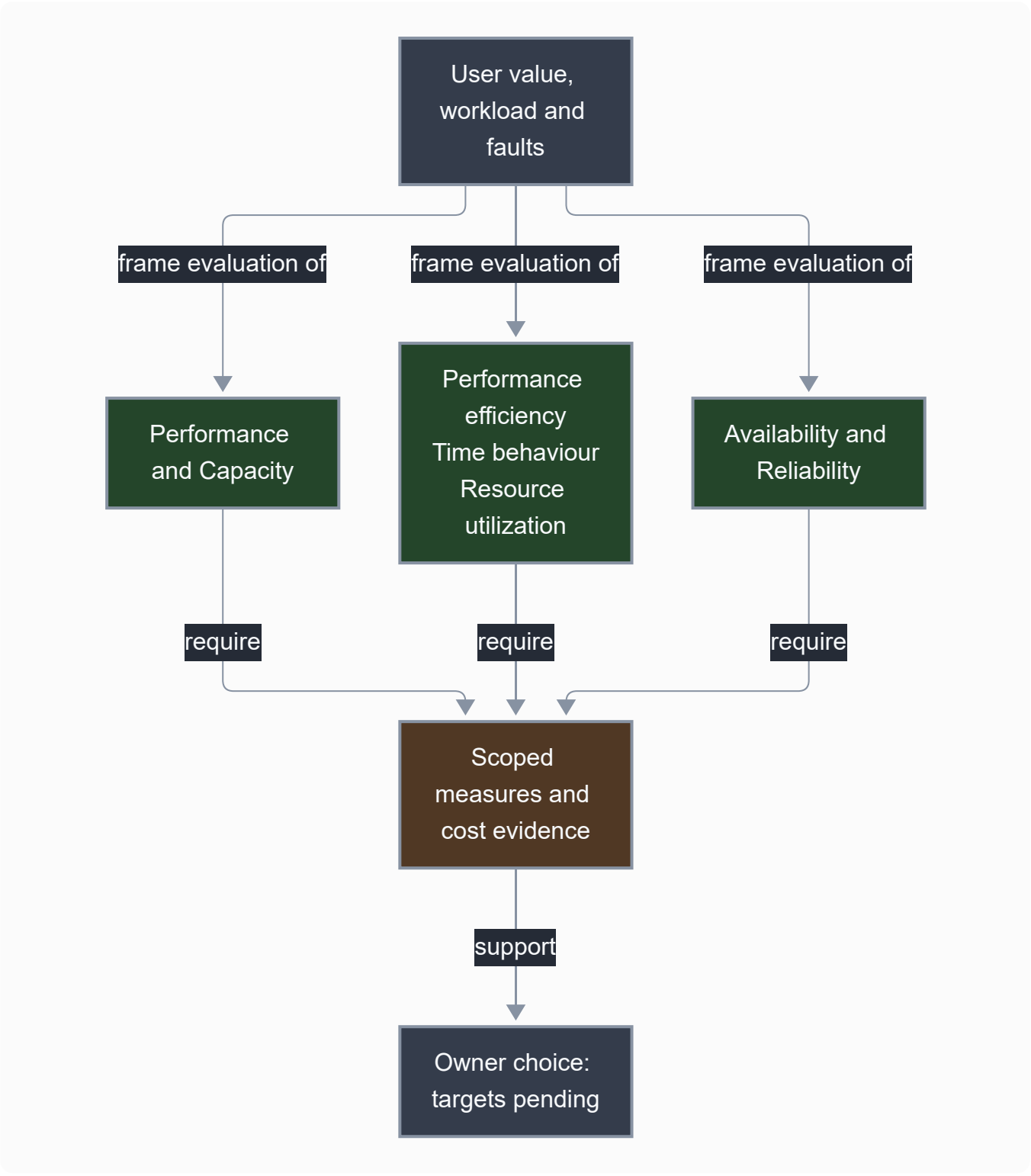
Quality attributes: executive narrative

Document review: **pending** after the interaction additions. Earlier editorial approvals remain historical; semantic, lifecycle and target states retain their separate scope.

This narrative helps decision-makers frame which user journey to evaluate, under what demand, and which correctness and cost conditions matter. G01 illustrates the reasoning; it reports no measured TDP result or approved commitment. G02 extends that reasoning to faults and recovery. The Performance comparison, Reliability comparison and their guides provide the canonical explanations and worked calculations.

Conceptual overview



Text equivalent: User value and workload frame evaluation of Performance and Capacity, together with Performance efficiency, Time behaviour and Resource utilization. Scoped measures and cost evidence support an owner decision; targets, budgets and priorities remain unapproved. This is a decision map, not a quality hierarchy. The same decision structure includes Availability and reliability under stated fault and recovery conditions.

Provenance: editorial synthesis of the linked approved local concepts and synthetic G01 and G02 examples discussed below. The arrows organize evaluation and proposed owner decisions, not domain

membership. No backend or external framework source is newly verified; the recorded source limits remain applicable. No approved TDP priority, budget or performance target is established.

Value and decisions

Predictable response and sufficient capacity can protect users from delay and support demand, while efficient resource use can limit operating expense. Insufficient quality can cause waiting, abandoned work, exhausted resources and additional incident effort. No TDP target, priority, owner, budget or deadline is approved. The owner should choose the important journey, workload, acceptable risk and cost envelope before selecting targets. Use the prioritization rubric.

Performance and capacity

Performance and capacity need workload-specific decisions: faster responses may require more capacity or expense. Response time, latency and throughput answer different questions; a single average does not describe a user's wait. Channel capacity limits communication and is not delivered throughput. The illustrative scenario shows how an owner could make a decision measurable without committing TDP.

Response time measures an explicitly bounded interval; it describes the whole user wait only when those boundaries include it. Latency measures delay between specified events and may cover only part of the journey, so that interval alone cannot promise the whole response. Throughput counts successful work and informs demand planning; meeting it while users wait too long or requests fail is insufficient. These are measures rather than independent quality attributes.

Efficiency and timing

Performance efficiency, time behaviour, and resource utilization expose trade-offs between speed and CPU, memory and bandwidth costs. Caching can improve waiting time but adds operational and consistency costs; scheduling may defer work. Fast, Responsive, efficiency and responsiveness need a scoped measure before they can support a decision.

CPU pressure can increase compute costs or limit demand; memory pressure can cause failures or require larger instances; bandwidth pressure can delay transfers or require network investment. Each needs its own budget and workload evidence. Responsive and Fast are Q42 labels, not approved thresholds: ask whose interaction or task should be timely. The secondary labels efficiency and responsiveness also need an owner to resolve their meaning before they drive commitments.

Worked decision: what would count as success?

The shared G01 example concerns a hypothetical project-information read service. It is not observed TDP architecture or a benchmark. All following values are synthetic and all criteria are illustrative and unapproved. The family comparison explains which of the five concepts answers each decision question.

G01 evaluates 50 requests/s against timing, correct authorized work, errors and stated resource limits together. Its synthetic observations cover a ten-minute hold including cache warming, after a one-minute ramp from an empty cache, with up to 100 concurrent users, 100,000 synthetic records and two service cores. Response time runs from client dispatch to complete receipt, excluding request preparation and result presentation. The canonical scenario and results retain the full illustrative criteria and method. G01-C varies demand in separate runs; G01-E compares caching alternatives under equivalent conditions.

Evidence question	Synthetic observation	Decision consequence
Is work timely and correct at the chosen load?	G01: Run A meets all illustrative criteria. Run B meets the 49 correct authorized requests/s criterion, but fails timing, errors and CPU.	A passing throughput counter cannot offset the other failures. Investigate them before proposing acceptance.
What demand can be sustained?	G01-C: separate runs pass at 40 and 50 requests/s; the 60 requests/s run fails despite completing more correct work per second.	Refine the tested boundary and assess uncertainty before choosing operating headroom.
Does caching justify its cost?	G01-E: the treatment returns the same 29,850 correct reads with 40% less service CPU per read and 50% more peak service memory.	Compare resource savings with memory, invalidation work and operating cost. No monetary saving is established.

> **Example: a percentile is not every user's wait.** In synthetic G01 Run A, successful-response p95 is 420 ms, while 450 successful responses take 900 ms. Errors are reported separately. The measured interval also excludes request preparation and result presentation. See the population and measurement boundaries.

> **Example: a tested load is not an operating limit.** Synthetic G01-C passes at 50 requests/s and fails at 60. Intermediate rates remain untested; choosing operating headroom requires further evidence and an owner decision. The capacity variant establishes neither an exact maximum nor a safe production limit.

Caching choice and consequences

Keeping no cache avoids invalidation machinery but repeats dependency work; in the synthetic comparison it fails the CPU limit. A bounded TTL cache can reduce work, but expiry permits stale reads. The example's TTL-only alternative returns 120 stale results and fails the zero-stale criterion even though its p95 is 400 ms and its error fraction is under 1%. A cache coordinated with updates passes the illustrative comparison, but adds memory, invalidation, recovery and diagnosis costs. It still requires tests of races, permission changes and missed updates. See the canonical alternatives and the technical evaluation.

> **Example: lower CPU use does not establish a financial saving.** Synthetic G01-E reduces service CPU per correct read by 40%, while peak service memory rises by 50%. Dependency costs, invalidation work and operating effort still require evaluation. See the controlled comparison; its illustrative correctness and resource criteria remain unapproved.

If this hypothetical journey proves relevant, the owner would first decide whether to investigate invalidation-aware caching and which correctness, demand and cost conditions should govern that investigation. Applicability, freshness semantics, targets, headroom, budget, priorities and owner are all pending. These examples justify further comparison; they authorize no implementation. A scenario checks a system against expectations; the baseline/treatment comparison checks a technique's effect. The quality attribute itself is not the intervention.

Availability and reliability

The Reliability and operations family adds the question of continued useful service. An interruption can block users and delay decisions; incorrect restoration can lose acknowledged work and create rework. The canonical G02 guidance distinguishes the quality questions. The hypothetical service, all following observations and numerical criteria are synthetic and unapproved for TDP.

Decision question	Business and delivery consequence	Evidence and choice requested
Availability: when is service needed?	More usable time may protect workflows; spare capacity and monitoring add recurring cost	Define required-use hours, eligible users and the usable-service predicate; uptime is seconds meeting it, downtime is seconds failing it
Reliability: how long must correct service continue?	Frequent interruptions undermine trust even when recovery is fast; prevention and longer trials cost delivery time	Define service failures and observation period. MTBF uses uptime exposure per failure, not a promise about the next incident
Fault tolerance: which faults may interrupt work?	Redundancy can preserve work but adds capacity, coordination and testing cost	Select tolerated faults and any permitted degradation; continuity during one stopped instance says nothing about a shared-store failure
Recoverability: what must be restored?	Fast restart can still leave lost work; restore rehearsals need staff, storage and delivery time	Choose time and data-loss bounds together; require reconciliation of acknowledged updates and valid access
Robustness: how should abnormal requests behave?	Controlled rejection protects valid work; overly restrictive validation can block legitimate users	Agree abnormal-input boundaries and error handling, then check correct valid work and no partial updates
Faultlessness and historical Maturity	Normal-operation evidence helps expose defects; neither a clean trial nor a new edition name guarantees defect absence	Review the historical meaning before adopting it; keep the two local concepts separate and avoid confusing product behavior with process maturity

G02 worked decision

In G02's 3,600 s required-use window, 10 valid reads/s produce 36,000 offered reads. One instance stops at t=1,200 s. Run A loses 20 s and 200 reads; its 3,580 / 3,600 = 99.444…% availability exceeds the illustrative 99% criterion, and 20 s restoration meets the illustrative 30 s bound. All 60 acknowledged updates remain correct, with zero duplicate effects or unauthorized results. Run B loses 80 s and 800 reads, giving 97.777…% availability and missing the restoration bound even though all updates survive. In B, the test controller holds the update scheduled during the outage until service returns; this example assumes no service-side queue. These are separate synthetic runs; neither was executed against TDP.

> **Example: acceptable recovery can still interrupt work.** Run A meets G02's time and data criteria but fails a stricter zero-failed-read continuity requirement. Strict variant C preserves all 36,000 reads during the instance fault, at a separately evaluated redundancy cost. Choose the actual business tolerance before choosing a mechanism. See the canonical results.

For measurement planning, a separate synthetic sample has 35,800 s uptime, 200 s downtime and two failures: MTBF is 35,800 / 2 = 17,900 s (about 4.97 h). It is an exposure-based sample, not an incident timetable. Time availability and request success have different denominators and can diverge when demand varies. Unknown monitoring intervals must stay visible. The measurement contract explains these limits.

G02-R separately checks 100 malformed requests alongside valid work. Controlled rejection of all 100 with no corruption passes its example; a crash on one request fails robustness even if aggregate availability still exceeds 99%. Faultlessness uses a no-injection normal-operation control, not this abnormal traffic. Neither check confirms the historical Maturity definition.

Dependable/dependability is a broad reading aggregate across trust concerns, including security and safety, not another leaf or a combined score. Available, Recoverable, Fault-tolerant, fault-tolerance and Robust retain scoped related mapping proposals. Consistent could refer to data, behavior or policy; its

meaning requires an owner decision. Security's availability concern uses the same local identity. See terminology limits; security and safety remain distinct concerns.

Use the existing value/risk rubric to select a real journey. The owner must choose fault coverage, hours of use, data-loss/freshness rules, interruption/recovery thresholds, budget and responsibilities. A failover investigation is reasonable only if the journey and interruption risk justify its capacity and operational cost; no investment, priority, service level or implementation is approved here. The technical evaluation compares mechanisms and rejection criteria. ISO full-text definitions and the maturity/faultlessness equivalence remain unverified; Q42 tags are reading aids.

Sources and further reading

The canonical G02 source record retains consultation dates and access limits. Primary publisher destinations:

- ISO/IEC 25010:2023 catalogue: publication metadata, not full-standard consultation.
- Historical ISO/IEC 25010:2011 catalogue: withdrawn edition; its labels do not establish current equivalence.
- arc42 Q42 model: its own reading tags; ISO attributions remain secondary evidence.
- SEI scenario method, CMU/SEI-2003-TR-004: historical six-component method, not measured TDP behavior.

Broader quality and lifecycle

The nine Q42 dimensions are overlapping prompts: efficiency, flexibility, maintainability, operation, reliability, safety, security, suitability and usability can compete or reinforce each other. Do not approve speed in isolation. Runtime consequences and development feedback both matter; the generated lifecycle groups record proposed individual assessments. The technical narrative examines verification and costs.

For a performance decision, **Efficient** prompts a cost comparison; **Flexible** asks whether the choice can adapt to changing demand; **Maintainable** weighs future change and diagnosis costs; and **Operable** asks whether staff can observe and control it. **Reliable** asks whether service stays correct under load; **Safe** considers harm from late outcomes; **Secure** retains necessary controls; **Suitable** checks the result against the real purpose; and **Usable** considers whether waiting and feedback allow people to complete their work. These are Q42's overlapping tags, not nine additional local attributes. Runtime evidence informs operational and investment decisions; development-time evaluation can expose risks earlier but costs delivery effort and cannot prove production results.

> **Example: testing before release still measures the service.** A hypothetical pre-release G01 evaluation would measure service behavior under load, not developer productivity or build duration. It would not confirm production behavior or the proposed lifecycle assignments. See the canonical explanation.

The following generated lifecycle groups reflect individual records in the published corpus. They can overlap, inherit nothing from the navigation family and do not classify all of TDP. `None.` means no published member in that group, not absence of the concern from TDP. Lifecycle reading explains the record states and independent confirmation requirements.

Security goals and decisions

Security can protect information and sustain trust, while disclosure, unauthorized changes or inability to attribute a decision can lead to recovery costs and lost confidence. The owner must decide which information and threats matter, which actions are permitted, and the acceptable risk, cost and delivery schedule. Authentication and authorization can support protection, but successful login alone does not

show that information is protected. Additional controls and evidence retention require operational effort and can slow legitimate work.

The security guide proposes a synthetic access scenario and criteria, not a TDP audit or approved target. Confidentiality, integrity, accountability, authenticity, non-repudiation and resistance need distinct evidence; Availability remains a shared concern. Safety is a distinct future topic.

Interaction and accessibility

An inaccessible task can prevent people from completing work and increase support costs. Decide which users and journeys matter, fund evaluation with affected people, and agree a remedy and acceptance criteria before release. A visually appealing interface does not prove accessibility or engagement. Compare the cost of inline guidance with separate help and the risk of errors or lost work; the G04 example is synthetic and its targets are unapproved.

A failed update can create rework, delay and support demand; an excluded group may be unable to participate at all. Decide which journeys and populations to evaluate, fund recruitment and accessible remediation, and agree how unresolved barriers affect release. Aesthetic preferences and engagement require separate questions; neither substitutes for access or successful recovery. The concept-specific observations show what each proposed decision needs as evidence. These are illustrative risks, not measured costs or approved TDP priorities.

Decision concern	Consequence to evaluate
Usability and interaction capability	A completed but difficult journey can increase rework and support demand; fund task evaluation before selecting a target.
Accessibility and disability access	An inaccessible path can prevent participation; decide how unresolved barriers affect release and fund remediation.
Inclusivity	A journey designed around one language or device may exclude other intended users; choose whom research must represent.
Recognizability and learnability	A hidden action or a task requiring repeated instruction can increase onboarding effort; fund discovery and repeat-task research.
Operability and error protection	Unusable controls or lost entries can prevent work and create rework; prioritize recovery and data preservation.
Assistance and self-descriptiveness	Unclear feedback shifts effort to support; compare the cost of clear messages and maintained help with user assistance needs.
Presentation, historical aesthetics and engagement	Appearance and continued involvement affect different decisions; consult intended users without treating attractiveness as evidence of access or retention. The two aesthetics identities remain distinct pending mapping review.

The split labels appropriateness and recognizability have no corroborated independent scoped meanings; do not budget them as separate qualities without owner review. The candidate labels aesthetic, attractive and operability retain proposed mappings only. These consequences are hypotheses, not measured costs.

Lifecycle reading groups

Explicit per-concept assessments; no parent inheritance. Assignments can overlap.

Runtime

- Accessibility — assessed; proposal: A barrier occurs when someone uses the interface with a particular technology; prototype evaluation can reveal it during development.

- Accountability — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Appropriateness recognizability — assessed; proposal: Recognition occurs when the person encounters the action; labels and placement can be evaluated in a prototype during development.
- Authenticity — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Availability — assessed; proposal: Running users need service readiness; pre-release evaluation tests the declared up-state predicate.
- Capacity — assessed; proposal: Running services encounter workload limits; development explores those limits before release.
- Confidentiality — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Disability access — assessed; proposal: The access path is encountered in operation and can be inspected in a representative prototype before release.
- Fault tolerance — assessed; proposal: Runtime service may continue during faults; development checks the fault model and remaining capacity.
- Faultlessness — assessed; proposal: Normal runtime operation can be observed; development evaluates representative normal traces before release.
- Inclusivity — assessed; proposal: Participation depends on real contexts of use; research with a varied sample can identify exclusions during design.
- Integrity — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Interaction capability — assessed; proposal: Discovery, control and recovery occur in use; an end-to-end prototype can expose a broken transition before release.
- Interface aesthetics and presentation — assessed; proposal: Presentation is perceived when the interface is used; mockups can be compared before implementation, subject to their fidelity.
- Learnability — assessed; proposal: Learning and recall occur across use; repeated prototype tasks can test the proposed design during development.
- Non-repudiation — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Operability (interaction) — assessed; proposal: Control is exercised at runtime; a representative interface can be exercised with declared input methods before release.
- Performance efficiency — assessed; proposal: Running services consume resources to deliver work; development compares the cost of candidate designs.
- Performance — assessed; proposal: Running services experience workload and timing; development evaluates response under representative load.
- Recoverability — assessed; proposal: Runtime recovery restores service and data; development rehearses restoration and validates recovery procedures.
- Reliability — assessed; proposal: Running services face interruptions over time; development evaluates failure histories and representative continuous operation.
- Resource utilization — assessed; proposal: Running services consume finite resources; development measures consumption against proposed budgets.
- Robustness — assessed; proposal: Running services encounter abnormal input; development explores boundaries and verifies controlled rejection.
- Security — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Self-descriptiveness — assessed; proposal: The interface communicates its state during use; message comprehension can be checked with a prototype in development.
- Time behaviour — assessed; proposal: Running services exhibit response distributions; development measures them under controlled conditions.
- Usability — assessed; proposal: The completed task and reported effort are experienced during use; a prototype can expose failures before release.
- User assistance — assessed; proposal: Help is consumed during the task; its discoverability and usefulness can be evaluated before release.
- User engagement — assessed; proposal: Involvement is experienced over use; a realistic prototype session can provide tentative design feedback, not a population estimate.

- User error protection — assessed; proposal: Errors and recovery occur during use; controlled invalid submissions can test the design during development.
- User interface aesthetics — assessed; proposal: Perception arises in use; representative mockups permit tentative comparison during development. This local assessment does not assign an ISO lifecycle.

Development time

- Accessibility — assessed; proposal: A barrier occurs when someone uses the interface with a particular technology; prototype evaluation can reveal it during development.
- Accountability — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Appropriateness recognizability — assessed; proposal: Recognition occurs when the person encounters the action; labels and placement can be evaluated in a prototype during development.
- Authenticity — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Availability — assessed; proposal: Running users need service readiness; pre-release evaluation tests the declared up-state predicate.
- Capacity — assessed; proposal: Running services encounter workload limits; development explores those limits before release.
- Confidentiality — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Disability access — assessed; proposal: The access path is encountered in operation and can be inspected in a representative prototype before release.
- Fault tolerance — assessed; proposal: Runtime service may continue during faults; development checks the fault model and remaining capacity.
- Faultlessness — assessed; proposal: Normal runtime operation can be observed; development evaluates representative normal traces before release.
- Inclusivity — assessed; proposal: Participation depends on real contexts of use; research with a varied sample can identify exclusions during design.
- Integrity — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Interaction capability — assessed; proposal: Discovery, control and recovery occur in use; an end-to-end prototype can expose a broken transition before release.
- Interface aesthetics and presentation — assessed; proposal: Presentation is perceived when the interface is used; mockups can be compared before implementation, subject to their fidelity.
- Learnability — assessed; proposal: Learning and recall occur across use; repeated prototype tasks can test the proposed design during development.
- Non-repudiation — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.
- Operability (interaction) — assessed; proposal: Control is exercised at runtime; a representative interface can be exercised with declared input methods before release.
- Performance efficiency — assessed; proposal: Running services consume resources to deliver work; development compares the cost of candidate designs.
- Performance — assessed; proposal: Running services experience workload and timing; development evaluates response under representative load.
- Recoverability — assessed; proposal: Runtime recovery restores service and data; development rehearses restoration and validates recovery procedures.
- Reliability — assessed; proposal: Running services face interruptions over time; development evaluates failure histories and representative continuous operation.
- Resource utilization — assessed; proposal: Running services consume finite resources; development measures consumption against proposed budgets.
- Robustness — assessed; proposal: Running services encounter abnormal input; development explores boundaries and verifies controlled rejection.
- Security — assessed; proposal: The proposed concern applies to operation and to design and pre-release evaluation; its scope remains subject to owner review.

- Self-descriptiveness — assessed; proposal: The interface communicates its state during use; message comprehension can be checked with a prototype in development.
- Time behaviour — assessed; proposal: Running services exhibit response distributions; development measures them under controlled conditions.
- Usability — assessed; proposal: The completed task and reported effort are experienced during use; a prototype can expose failures before release.
- User assistance — assessed; proposal: Help is consumed during the task; its discoverability and usefulness can be evaluated before release.
- User engagement — assessed; proposal: Involvement is experienced over use; a realistic prototype session can provide tentative design feedback, not a population estimate.
- User error protection — assessed; proposal: Errors and recovery occur during use; controlled invalid submissions can test the design during development.
- User interface aesthetics — assessed; proposal: Perception arises in use; representative mockups permit tentative comparison during development. This local assessment does not assign an ISO lifecycle.

Unassessed

- Maturity — unassessed; proposal: No lifecycle assignment is adopted before the historical meaning and scope are corroborated.
- Resistance — unassessed; proposal: The precise scope of resistance requires corroboration before assigning lifecycle relevance.

Assessed without assignment

None.

Presentation export

Download PDF. Ordinary builds are not export-verified; run the documented export command and check the export manifest for generation time, input digest and file hashes.

References and online details

These online destinations require a connection. Portal pages may change after this PDF was generated; the footer identifies this export. URLs remain readable when printed.

1. Performance comparison: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/performance-and-capacity/#compare-the-questions
2. Reliability comparison: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reliability-and-operations/#compare-the-questions
3. recorded source limits: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#sources-and-scope
4. prioritization rubric: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#prioritization
5. Performance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/performance/
6. capacity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/capacity/
7. Response time, latency and throughput: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#timing-and-throughput
8. Channel capacity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#resources-and-channel-capacity

9. illustrative scenario: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#g01-scenario
10. Performance efficiency: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/performance-efficiency/
11. time behaviour: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/time-behaviour/
12. resource utilization: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/resource-utilization/
13. Fast, Responsive, efficiency and responsiveness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#aliases
14. canonical scenario and results: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#g01-worked-results
15. G01-C: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#g01-c-capacity-variant
16. G01-E: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#g01-e-efficiency-variant
17. population: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#percentiles-and-populations
18. measurement boundaries: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#request-journey-and-measurement-boundaries
19. canonical alternatives: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#caching-alternatives-and-verification
20. technical evaluation: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/technical/#controlled-evaluation-of-g01
21. Reliability and operations family: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reliability-and-operations/
22. G02 guidance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/reliability-guide/
23. Availability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/availability/
24. Reliability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/reliability/
25. Fault tolerance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/fault-tolerance/
26. Recoverability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/recoverability/
27. Robustness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/robustness/
28. Faultlessness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/faultlessness/
29. Maturity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/maturity/
30. canonical results: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/reliability-guide/#g02-worked-results
31. measurement contract: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/reliability-guide/#measures-and-boundaries
32. terminology limits: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/reliability-guide/#aliases-and-unresolved-meanings
33. technical evaluation: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/technical/#availability-and-reliability
34. canonical G02 source record: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/reliability-guide/#sources-and-scope
35. ISO/IEC 25010:2023 catalogue: https://www.iso.org/standard/78176.html
36. Historical ISO/IEC 25010:2011 catalogue: https://www.iso.org/standard/35733.html

37. arc42 Q42 model: https://quality.arc42.org/articles/arc42-quality-model
38. SEI scenario method, CMU/SEI-2003-TR-004: https://www.sei.cmu.edu/documents/704/2003_005_001_14213.pdf
39. nine Q42 dimensions: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#q42-dimensions
40. technical narrative: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/technical/
41. canonical explanation: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/performance-guide/#lifecycle
42. Lifecycle reading: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/lifecycle/
43. security guide: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/security-guide/#g03-scenario
44. G04 example: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/interaction-guide/#g04-scenario
45. concept-specific observations: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/reference/interaction-guide/#concept-specific-observations
46. Accessibility: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/accessibility/#lifecycle-assessment
47. Accountability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/accountability/#lifecycle-assessment
48. Appropriateness recognizability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/appropriateness-recognizability/#lifecycle-assessment
49. Authenticity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/authenticity/#lifecycle-assessment
50. Availability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/availability/#lifecycle-assessment
51. Capacity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/capacity/#lifecycle-assessment
52. Confidentiality: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/confidentiality/#lifecycle-assessment
53. Disability access: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/disability-access/#lifecycle-assessment
54. Fault tolerance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/fault-tolerance/#lifecycle-assessment
55. Faultlessness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/faultlessness/#lifecycle-assessment
56. Inclusivity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/inclusivity/#lifecycle-assessment
57. Integrity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/integrity/#lifecycle-assessment
58. Interaction capability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/interaction-capability/#lifecycle-assessment
59. Interface aesthetics and presentation: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/interface-aesthetics-presentation/#lifecycle-assessment
60. Learnability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/learnability/#lifecycle-assessment
61. Non-repudiation: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/non-repudiation/#lifecycle-assessment
62. Operability (interaction): https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/operability-interaction/#lifecycle-assessment
63. Performance efficiency: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/performance-efficiency/#lifecycle-assessment

64. Performance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/performance/#lifecycle-assessment
65. Recoverability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/recoverability/#lifecycle-assessment
66. Reliability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/reliability/#lifecycle-assessment
67. Resource utilization: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/resource-utilization/#lifecycle-assessment
68. Robustness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/robustness/#lifecycle-assessment
69. Security: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/security/#lifecycle-assessment
70. Self-descriptiveness: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/self-descriptiveness/#lifecycle-assessment
71. Time behaviour: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/time-behaviour/#lifecycle-assessment
72. Usability: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/usability/#lifecycle-assessment
73. User assistance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/user-assistance/#lifecycle-assessment
74. User engagement: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/user-engagement/#lifecycle-assessment
75. User error protection: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/user-error-protection/#lifecycle-assessment
76. User interface aesthetics: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/user-interface-aesthetics/#lifecycle-assessment
77. Maturity: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/maturity/#lifecycle-assessment
78. Resistance: https://zealous-bay-01d7e9503.5.azurestaticapps.net/architecture/qualities/resistance/#lifecycle-assessment