

# Latch-X: Availability & Risk Modeling with Bayesian Networks and Monte Carlo Simulation

Version v5

Peter Kováč

31 Aug 2025

# Contents

1. Introduction . . . . .	1
2. Motivation & Context . . . . .	1
3. Modeling Language (YAML) . . . . .	1
3.1 Component Semantics . . . . .	1
3.2 Dependency Semantics . . . . .	2
3.3 Distribution & Units . . . . .	2
4. Latch Nodes: Time-Bounded Transitions in a Static BN . . . . .	2
4.1 Modes . . . . .	2
4.2 Use Cases . . . . .	2
4.3 Composition . . . . .	2
5. Analysis Engines . . . . .	2
5.1 BN Engine (steady-state & mission) . . . . .	2
5.2 Monte Carlo Engine (discrete-event) . . . . .	3
5.3 Shared Result Schema . . . . .	3
6. Graph Specification (for Reproducible Diagrams) . . . . .	3
7. Validation & Integrity . . . . .	4
8. Case Study: Cross-DC Active/Standby Failover with Latch . . . . .	4
9. Risk-Aware Modeling via do: . . . . .	5
10. RTO Targets & MTD . . . . .	5
11. Implementation Overview . . . . .	5
12. Practical Defaults & Heuristics . . . . .	5
13. Limitations & Future Work . . . . .	6
14. Author's Note & Contributions . . . . .	6
15. Licensing and Trademark . . . . .	6
16. Selected References & Further Reading . . . . .	6
Appendix A: Minimal Valid Model . . . . .	7
Appendix B: Parameter Units & Conversions . . . . .	7
Appendix C: Latch Pattern Snippets . . . . .	7

## Abstract

Latch-X unifies Fault-Tree intuition, Bayesian-Network (BN) inference, and discrete-event Monte Carlo (MC) simulation into a single declarative YAML language. The framework introduces two practical abstractions: (1) a **do**: **causal relation** linking human/automation/disaster risks to technical components, and (2) a **latch node**, a one-shot, time-bounded transition that captures delayed failover/detection events while preserving BN acyclicity. Latch-X couples these with a schema-validated model, integrity **hashblock** signing, and a consistent **graph specification** for reproducible analysis and publication-ready diagrams.

## Contents

### 1. Introduction

This work draws on established reliability methods—Fault Tree Analysis (FTA) and Reliability Block Diagrams (RBD)—together with practical operations experience. To capture causal structure, we adopt Bayesian Networks (BN) and retain familiar logic gates (**and**:, **or**:, **n\_of\_k**:). To express operational causes (human change, automation, environment) without burying them inside components, we use a simple causal link: **do**:.

The hardest problem to solve was representing **failover dynamics** in a static BN. After many iterations we introduced the **latch** component - a one-shot, time-bounded transition that succeeds if a sampled delay is within a deadline. It closed the gap cleanly and gave Latch-X its name.

To share this beyond our own scripts, we packaged the approach as a **SaaS**: a declarative YAML feeds a stateless **core** (JSON/SVG out), with a light **UI** and an **AI assistant** to help non-specialists. The result aims to be accessible to architects and engineers designing for availability, risk, and downtime.

---

### 2. Motivation & Context

High-availability systems blend physical infrastructure, virtual platforms, networks, software stacks, and operations. Traditional tools (FTA, RBD, Markov) struggle to simultaneously express **causal risk propagation**, **delayed failovers**, and to **switch** between steady-state analytics and time-series simulation. Latch-X addresses this gap - a single, signed YAML becomes the source of truth for **causal structure**, **timing behavior**, **analysis settings**, and **graphics**.

---

### 3. Modeling Language (YAML)

A Latch-X model is a single YAML document with six core blocks **plus top-level metadata**:

- **components**: typed nodes (**normal**, **logical**, **latch**, **illogical**) with **mttf**, **mttr**, or **prob**; optional distributions (**exp** default; **delta**, **norm/lognorm** with **sigma**), **category**, **tags**, and **cost** (on normal nodes).
- **dependencies**: one dependency **type per target** from {**do**, **and**, **or**, **n\_of\_k**}.
- **root**: the top-level availability / reliability node.
- **simulation\_settings**: **repair\_enabled**, **simulation\_time**, optional **simulation\_runs** and **targets** (**availability**, **rto**).
- **graph\_settings**: **layout** (**fta** | **rbd** | **dag**), **label toggles**, **color maps** by category.
- **hashblock**: integrity metadata (**algo**, **hash**, **sig\_algo**, **signature**).
- **Top-level metadata**: **schema\_version**, **id**, **name**, **description**, **version**, **author**, **updated\_at**.

#### 3.1 Component Semantics

- **Normal** — repairable or mission components, parameterized by **mttf+mttr** *or* direct **prob** (**availability** / **reliability** over the analysis window).

- **Logical** — no intrinsic failure; defines structure via `and`, `or`, `n_of_k`.
- **Latch** — *one-shot* time-bounded transition triggered by parents. Use **either** `prob` or `mttf+max_delay`. No `mttr`.
- **ilogical** — inverted logical behavior (reserved / experimental).

### 3.2 Dependency Semantics

- **do**: causal link; parent  $\rightarrow$  child; used for structural chains and to inject **risk** causes into normal targets.
- **and**: serial path on a logical target.
- **or**: redundancy on a logical target (instantaneous switchover).
- **n\_of\_k**: quorum with integer `n` on a logical target.

### 3.3 Distribution & Units

- Default failure/repair distribution is **exponential** unless specified. Hours are the canonical unit for `mttf`, `mttr`, `max_delay`, `simulation_time`, and `rto`.
- `delta` is available for repair times (`mttr_dist`) to model deterministic restorations; when using `norm/lognorm` for either failure or repair, provide `sigma`.

---

## 4. Latch Nodes: Time-Bounded Transitions in a Static BN

**Problem:** Standard BNs lack explicit timing; Markov models add state explosion.

**Idea:** Represent *delayed* transitions as **latch nodes** that are triggered on parent failure and **succeed** if the sampled  $delay \leq max\_delay$ . Otherwise, the latch **fails**, propagating outage.

### 4.1 Modes

- **Probability mode:**  $prob = P(success)$  when activated.
- **Time mode:** Exponential delay with mean `mttf`:  
 $P(delay \leq max\_delay) = 1 - e^{-max\_delay/mttf}$ . Success if  $delay \leq max\_delay$ .

### 4.2 Use Cases

- Active/standby failover under a deadline or with uncertainty
- Operator detection delays
- Circuit breakers / switchover guards

### 4.3 Composition

Multi-stage processes are composed by chaining latches (each one-shot), optionally combined with logical gates to represent prerequisites (e.g., “failover AND standby present”).

---

## 5. Analysis Engines

### 5.1 BN Engine (steady-state & mission)

#### Why Bayesian Networks for availability & risk?

BNs provide an explicit, causal factorization of the joint distribution across components and risks while remaining acyclic and relatively compact compared to state-exploding Markov chains. Also their acyclic

nature is ideal for FTA representation. They let us: (a) represent structural logic (and:, or:, n\_of\_k:) as deterministic CPDs, (b) inject operational causes with do: parents (c) answer what-if queries by conditioning on evidence, and (d) compute diagnostic measures directly from the model without ad-hoc scripts.

#### Inference modes

- **Availability (repairable):** CPDs from `mttf/mttr` (or `prob`) yield exact marginals  $P(\text{node} = UP)$  for all nodes. CPDs use steady-state probabilities  $A = mttf / (mttf + mttr)$
- **Reliability (mission):** With `repair_enabled: false`, CPDs use survival over `simulation_time` (mission duration) from the chosen distributions.

#### Diagnostic analytics exposed in Latch-X

All computed by exact inference (Variable Elimination) on the BN:

##### 1) Impact analysis (marginal contribution)

$$Impact(X) = P(Root = UP) - P(Root = UP | X = DOWN)$$

Larger values mean observing  $X = DOWN$  most reduces service availability.

##### 2) Root-cause posterior (when the service is down)

$$P(X = DOWN | Root = DOWN) = P(X = DOWN, Root = DOWN) / P(Root = DOWN)$$

A diagnosis view: given an outage, which components are most likely the cause?

##### 3) Sensitivity explorer (what-if with evidence)

For any target set  $T$  and evidence  $E$  (e.g., a risk present, a standby absent), we query  $P(T = UP | E)$  across evidence configurations to see which controls/assumptions move availability the most.

Because the model uses explicit do: causes, conditioning generally aligns well with real world scenarios (e.g., “patch error occurred”).

## 5.2 Monte Carlo Engine (discrete-event)

- Event-driven simulation over `simulation_time`, using a time-ordered event queue.
- Models failure, repair, and latch delays as timestamped events with per-node distributions.
- Produces availability distributions (histograms, percentiles) and breach probabilities for SLA/RTO/MTD.
- Complements the BN engine: BN yields exact steady-state/mission marginals; MC yields empirical distributions and confidence intervals.

## 5.3 Shared Result Schema

Both engines return: - Per-node availability (and reliability in mission mode)

- In other aspects engines are complementary.

## 6. Graph Specification (for Reproducible Diagrams)

### Nodes

- **normal:** rectangle, light blue fill, solid border
  - **logical:** rectangle, light blue fill, **dashed** border
  - **latch:** rounded rectangle, light gray fill, solid border
  - **root:** hexagon, light goldenrod fill, **bold** border
  - **risk category:** same shapes; colored via `graph_settings.category_colors`
- Note: root is not a node type in the schema; the hexagon is a visual highlight only.

### • Edges

- All edges are directed parent → child with arrowheads and labels: do/and use solid lines; or/n\_of\_k use dashed lines (label `n_of_k(n)`).

Layouts: **FTA** (top-to-bottom), **RBD** (left-to-right), **DAG** (force-directed). All export as **SVG** for reports.

---

## 7. Validation & Integrity

**Rules enforced by validator:** - Every dependency name must exist under components.

- Exactly **one** dependency type per target.
- Type discipline: and/or/n\_of\_k attach to **logical** targets; do is used also for **normal** targets.
- For latches: if mttf is provided, max\_delay is **required**.
- For normals: use **either** prob **or** mttf+mttr (not both).
- If norm/lognorm is selected for failure/repair, provide sigma.

**Integrity:** Models are signed with a **hashblock** {algo, hash, sig\_algo, signature} for content authenticity and change detection.

---

## 8. Case Study: Cross-DC Active/Standby Failover with Latch

Below is a compact pattern demonstrating a DB tier with standby and a time-bounded failover latch, then consumed by a virtual DB and an application service.

```
root: service
components:
  # logical aggregators
  service:      { type: logical }
  failover_path: { type: logical }
  db_virtual:   { type: logical }

  # primaries/standby + latch
  db_primary:   { type: normal, mttf: 24000, mttr: 1 }
  db_standby:   { type: normal, mttf: 24000, mttr: 1 }
  db_failover:  { type: latch,  mttf: 0.5, max_delay: 1.0 } # avg 30 min, 1 h deadline

dependencies:
  db_failover:
    do: [db_primary]           # latch activates on primary failure

  failover_path:
    and: [db_failover, db_standby] # need successful failover AND standby

  db_virtual:
    or: [db_primary, failover_path] # virtual DB is up via primary OR failover path

  service:
    do: [db_virtual]

simulation_settings:
  simulation_time: 8760
  repair_enabled: true
  targets: { availability: 0.999, rto: 4 }
  simulation_runs: 1000
```

**Interpretation:** BN provides exact steady-state availability; MC yields availability distribution and **P(recovery > 4 h)**. Adjust mttf, mttr, and latch timing to evaluate design variants (e.g., automated vs manual switchover).

---

## 9. Risk-Aware Modeling via do:

Operational risks (configuration error, patch error, automation mistake, datacenter events) are modeled as type: normal nodes with category: risk and linked to affected components via do:. Parameterize them with mttf/mttr to capture **frequency × duration** over the analysis window; use a single prob only for BN steady-state shortcuts when you do not need time-domain simulation.

```
components:
  patch_error: { type: normal, category: risk, mttf: 2000, mttr: 2 }
  service:     { type: logical }
  db_virtual:  { type: logical }

dependencies:
  db_virtual:
    do: [patch_error]    # risk injects into the technical layer
  service:
    do: [db_virtual]
```

---

## 10. RTO Targets & MTD

Set `targets.rto` to the operational objective (RTO) or business objective (MTD) (e.g., 4 h).

-  $P(\text{recovery} > RTO)$  — operational target breach

-  $P(\text{recovery} > MTD)$  — business hard-limit breach

You can evaluate **MTD** by temporarily setting `targets.rto = MTD`.

---

## 11. Implementation Overview

- **Core:** Stateless service; accepts the complete YAML and returns **JSON** (results) or **Graphviz SVG** (diagram).
  - **UI:** Session-based frontend with YAML editor, forms-based builder, and AI assistant.
  - **Preprocessing:** Auto-validation, rule enforcement, digital signing
  - **Extensibility:** Distribution families are pluggable; per-node dist selection is supported (default exp).
- 

## 12. Practical Defaults & Heuristics

- **Time base:** `simulation_time: 8760 h` (1 year) for availability; set mission durations for reliability with `repair_enabled: false`.
  - **Simulation runs:** 100–1 000 for exploration; 10 000+ for tight bounds.
  - **RTO:** Choose below business **MTD**; review both RTO and MTD breach probabilities from MC.
  - **Parameterization:** Use realistic mttf/mttr from incident data or vendor baselines; avoid mixing prob with mttf/mttr on the same node.
  - **Graph settings:** Prefer FTA for root-cause storytelling; RBD to show success paths; DAG for causality or larger graphs.
-

## 13. Limitations & Future Work

- **Distributions:** norm/lognorm require sigma; richer families (Weibull) are candidates.
  - **Latch semantics:** One-shot by design; multi-stage processes require composition.
  - **Scale:** Very large BNs trigger safety guards; future versions may add sparse inference and incremental caching.
  - **Integration:** Integrating with enterprise model repositories or CMDBs.
  - **Other Domains:** Other risk domains can be added (e.g., Cybersecurity).
  - **Prob-only nodes:** When only prob is provided, time-domain simulation treats nodes as static per run and does not model arrival/repair timelines; prefer  $mttf/mttr$  (or  $latch\ mttf/max\_delay$ ) when RTO/overlap effects matter.
- 

## 14. Author's Note & Contributions

This project assembles known techniques into a practical workflow and adds a few small, useful pieces that we found missing in everyday reliability work:

- **Causal modeling with BNs:** reuse of FTA-style gates inside a BN, plus a causal `do`: link to inject operational risks (human, automation, environmental) into technical components.
- **Latch node:** a time-bounded, one-shot transition for delayed failover/detection in a static BN; usable either via a direct success prob or via  $mttf + max\_delay$ .
- **SaaS packaging:** declarative **YAML** → stateless **core** (JSON/SVG) → session-based **UI**, with a simple graph specification and a **hashblock** for tamper evidence.

These are incremental, engineering-driven contributions intended to make reliability analysis more usable day to day.

---

## 15. Licensing and Trademark

Except where otherwise noted.

- **Licensing (whitepaper text and figures):** © 2025 Peter Kováč. Licensed under **Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0)**. See <https://creativecommons.org/licenses/by-nc/4.0/>.
  - **Commercial permissions:** For commercial use or reprints, contact [legal@latch-x.com](mailto:legal@latch-x.com).
  - **Scope:** This license applies to the whitepaper content only. **Software, source code, and any normative schema/specification are © 2025 Peter Kováč — All rights reserved — unless a separate license is expressly provided.** Any schema excerpts within this document are illustrative and do not grant rights to reproduce or implement the schema.
  - **Attribution:** Please cite as Kováč, P. (2025, August 31). *Latch-X: Availability & Risk Modeling with Bayesian Networks and Monte Carlo Simulation*. Whitepaper. **CC BY-NC 4.0**.
  - **Trademark:** Latch-X is a registered trademark in the European Union (EUTM No. 019183034). Outside the EU, Latch-X is used as an unregistered trademark; registration is planned. This license does not grant trademark or brand rights.
- 

## 16. Selected References & Further Reading

Core methods & textbooks

- Bertsekas, D. P., & Tsitsiklis, J. N. (2008). Introduction to Probability (2nd ed.). Athena Scientific.
- Bellot, D. (2016). Learning Probabilistic Graphical Models in R. Packt.
- Stewart, W. J. (2009). Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University Press.



- Stone, J. V. (2013). *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Sebtel Press.
- Ericson II, C. A. (2011). *Fault Tree Analysis Primer*.

### Practical HA & engineering

- Taylor, Z. (2013). *Designing High Availability Systems: DFSS and Classical Reliability Techniques with Practical Real-Life Examples*. Wiley.
- Smith, D. J. (2005). *Reliability, Maintainability and Risk* (7th ed.). Elsevier/Newnes.
- Bauer, E. (2010). *Design for Reliability: Information and Computer-Based Systems*. Wiley.
- Bauer, E. (2011). *Beyond Redundancy: How Geographic Redundancy Can Improve Service Availability and Reliability of Computer-Based Systems*. Wiley. (ASIN: B00HLFRA1O)
- Oggerino, C. (2001). *High Availability Network Fundamentals*. Cisco Press.

### Standards & handbooks

- IEC 61025 — *Fault Tree Analysis (FTA)*, latest edition.
- IEC 60300 series — *Dependability management* (selected parts).
- U.S. NRC (1981). *Fault Tree Handbook (NUREG-0492)*.

---

## Appendix A: Minimal Valid Model

```
root: top_component
components:
  top_component: { type: logical }
dependencies: {}
```

## Appendix B: Parameter Units & Conversions

- Hours are the canonical unit.
- 1 year  $\approx$  8 760 h.
- Typical availability to downtime conversion:  $downtime_h = (1 - A) \times simulation_time$ .

## Appendix C: Latch Pattern Snippets

### Probability mode

```
components:
  switchover_guard: { type: latch, prob: 0.97 }
```

### Time-based mode

```
components:
  db_failover: { type: latch, mttf: 0.5, max_delay: 1.0 }
```

### n\_of\_k shape (example):

```
db_quorum:
  n_of_k: { n: 2, inputs: [db1, db2, db3] }
```

---

*End of Whitepaper*