# Scaling Formal and Informal (Less Formal?) Reasoning

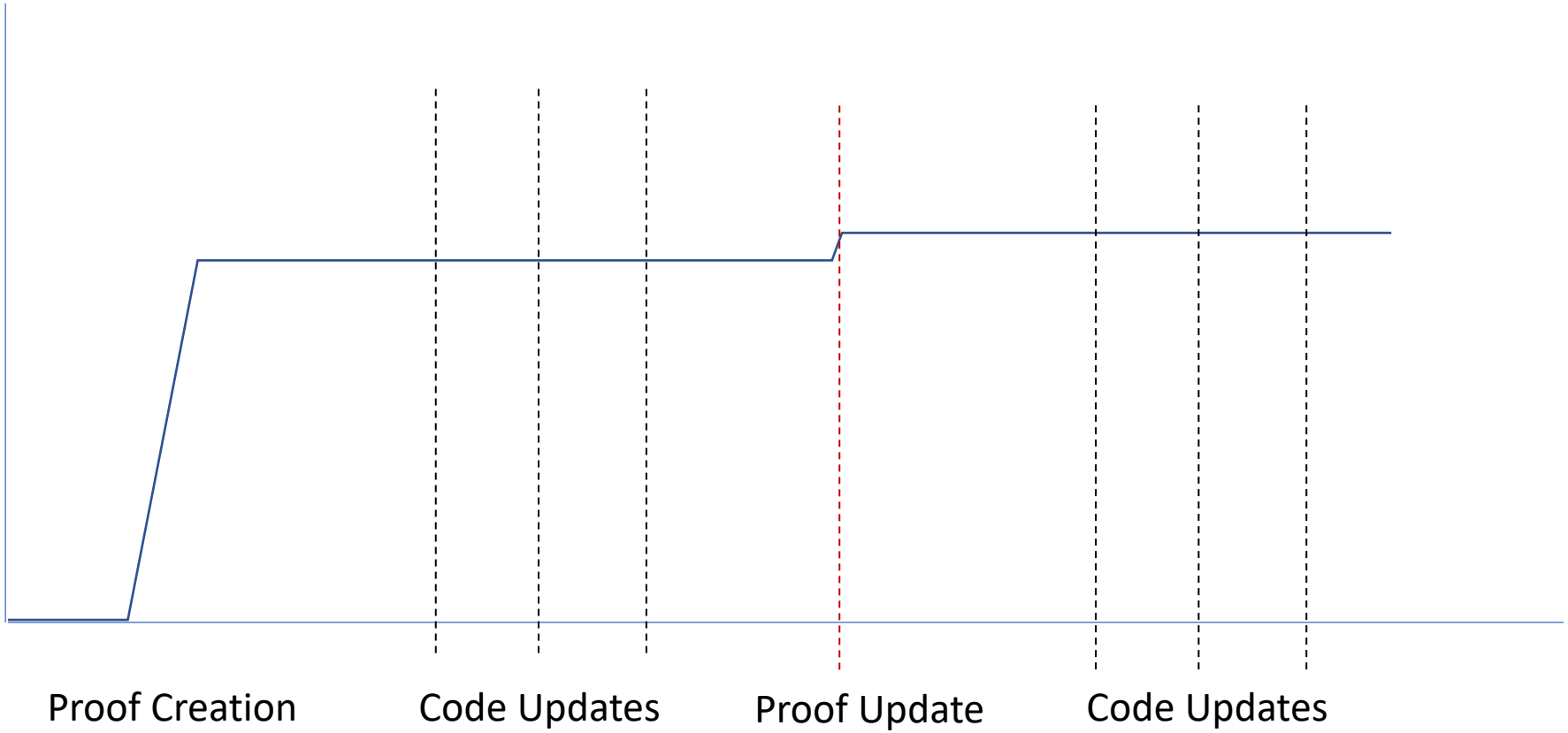
Stephen Magill

Galois, Inc / Muse Dev, Inc.

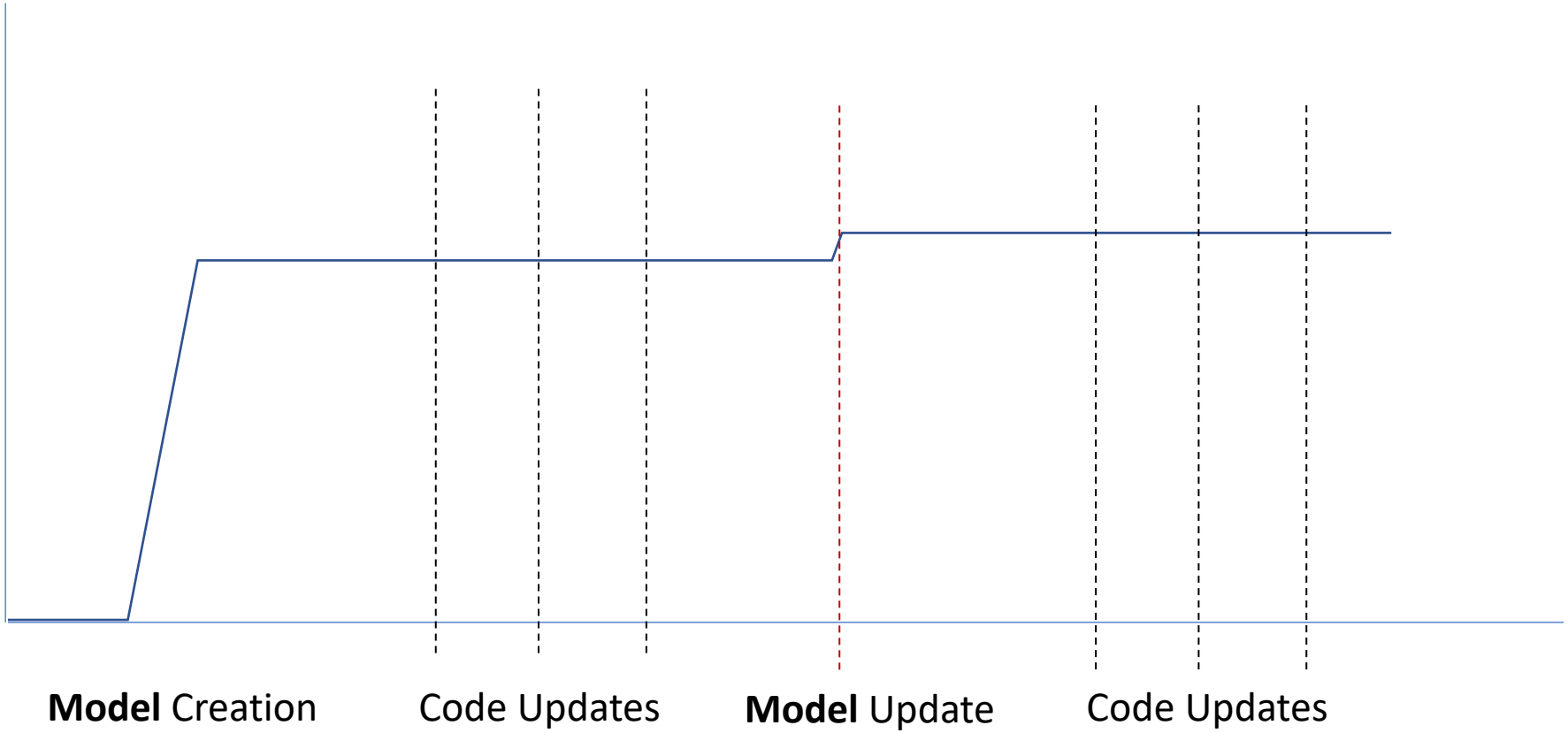
# Formal Analysis

- Sound functional correctness verification
- Relevant dimension of scale
  - Proof creation
  - Proof checking
  - Proof update
  - Program update
  - Code usage

# Effort Graph



# Less Formal



# Examples

- Functional Correctness Proofs for Amazon s2n
- Proofs for actively-developed cryptographic algorithms

# Less Formal Scaling

- Non-exhaustive formal methods based testing (PKCS11)
- Symbolic bugfinding methods (Muse Dev)
- Dimensions of scale
  - Lines of code (3.5B)
  - Commits per day (10s of thousands)

# Some Numbers

- Code size
  - 2B lines of code (Google)
  - 3.5B lines of code (unnamed financial sector company)
- Repository structure
  - 11k repositories
  - 1 repository with hundreds of projects
- Build time
  - 26 hours build time (different unnamed financial sector company)
    - 40 minutes when fully parallel!
  - “Trying to get builds under 10 minutes”

# Solutions

- Caching
  - Build artifacts
  - Analysis artifacts
  - Relevant Build-oriented Tools: Artifactory, Bazel, Nexus
- Incremental analysis
- “Minimal viable analysis”

# Inherent Sweet Spot for Each?

Are all practical industry-scale automated code analysis techniques necessarily unsound, while human-guided semi-automated analysis is the practical answer to cases where soundness is needed?