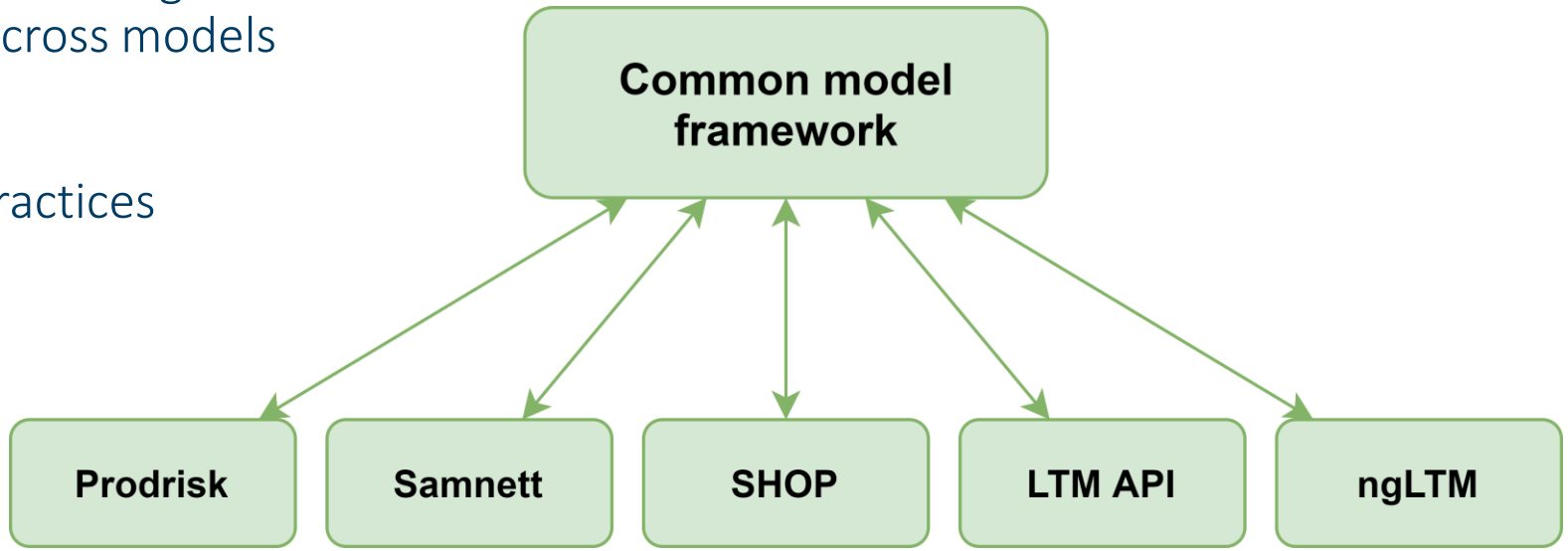
SINTEF
75 years

Common Framework

David Myklebust – Developer, ngLTM

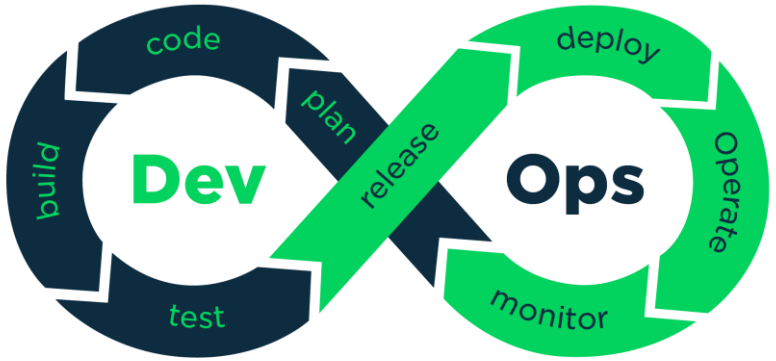
Common Model Framework

- Purpose:
  - Bring added value by utilizing solutions to common use cases across models
- Scope:
  - Development best practices
  - Design principles
  - Libraries



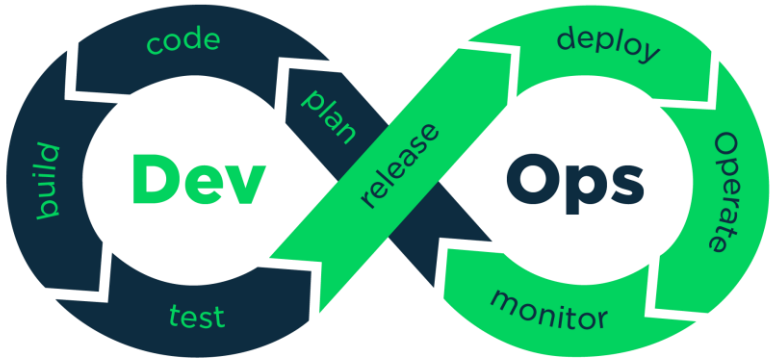
Common framework:
Development best practices

- Development cycle:
  - Source version control (Git)
  - Code reviews
  - Test automation and pipelines
- Documentation
- Software supply chain management

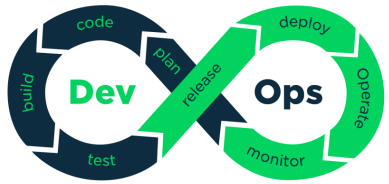


Common framework:
Git usage best practices

- Git branching strategy
  - Main branch
  - Feature branches
  - Merge requests & code review
  - Release tags
- Tools of the trade: GitLab
  - Git
  - Pipelines and runners
  - Code review tools
  - Package repository

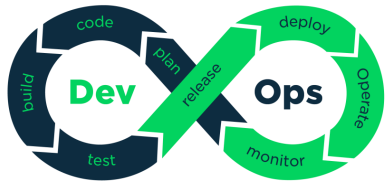


Common framework: Code reviews

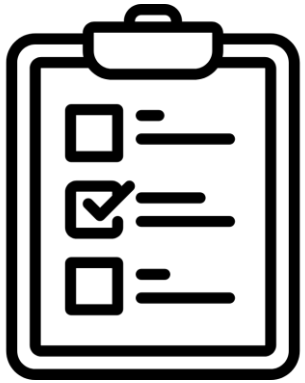


- Each code change is reviewed by a peer
- Integral to ngLTM project and adapted in other model projects
- Benefits:
  - Mistakes are caught early; minimizes their impact
  - **Greatly** increases code quality
  - Fosters **knowledge sharing** and **consensus building**

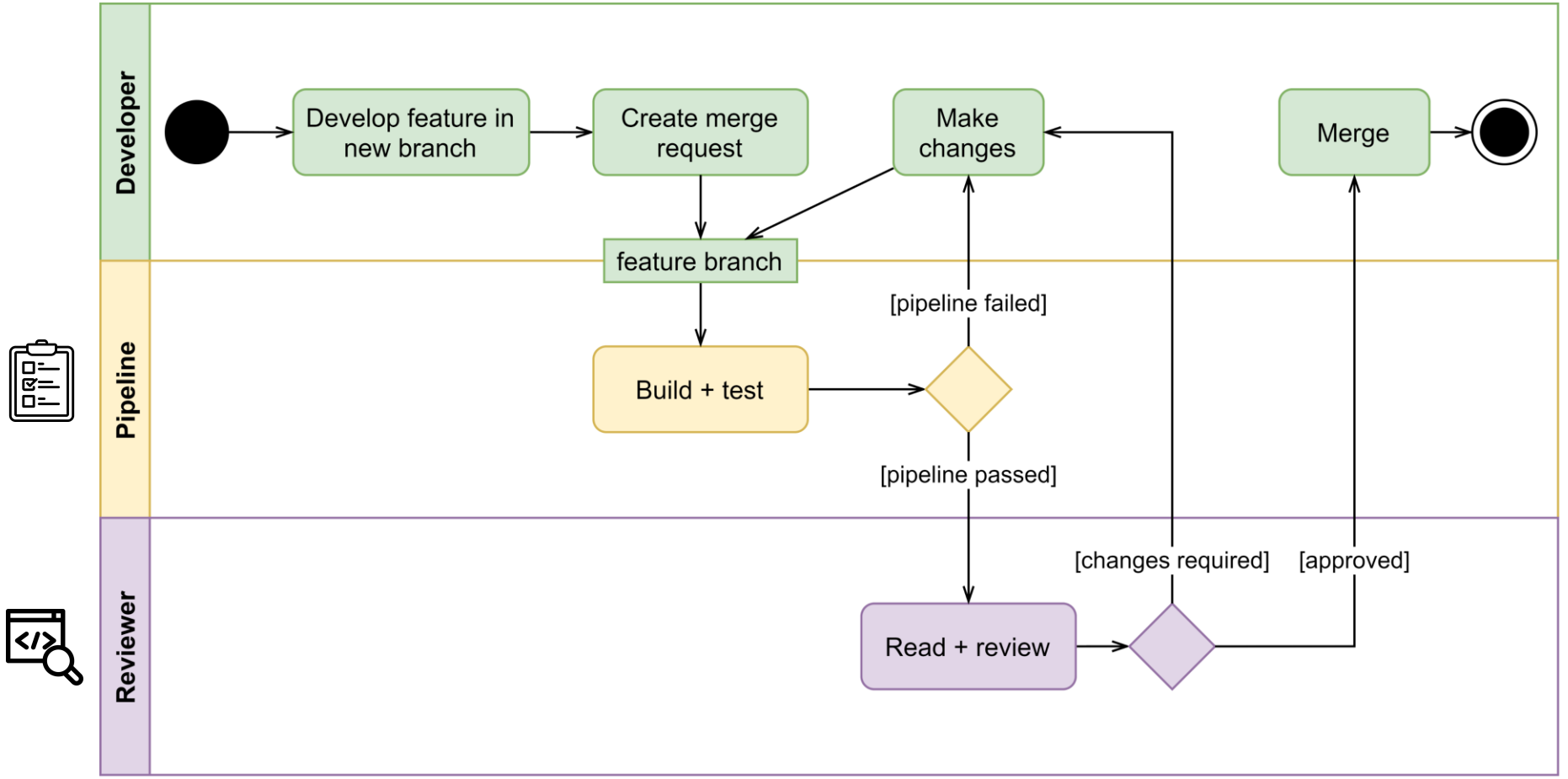
Common framework:
Test automation and pipelines



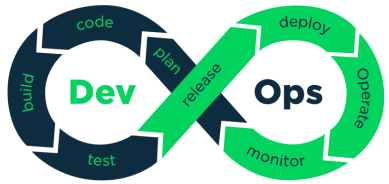
- Pipeline tasks:
  - **Build** libraries, binaries, packages, documentation
  - **Run tests** (unit-, integration-, and system tests)
  - **Code coverage**
  - **Static analysis and code formatting** («linting»)
  - **Vulnerability scanning**
  - **Deployment**
- Pipelines are part of the Merge Request process
  - Protect integrity of main branch: don't merge if pipeline fails
  - *The main branch must always be in a buildable and usable state*



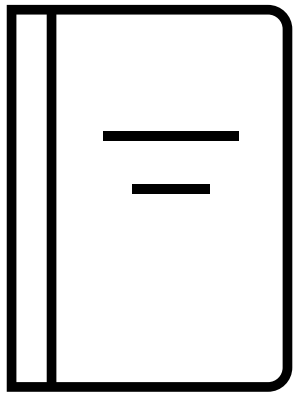
Common framework:
Development cycle



Common framework: Documentation

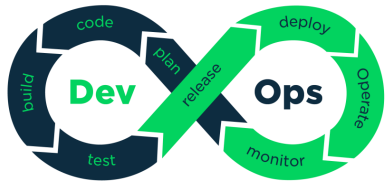


- Tools:
  - Command line tool to build user guide & API reference (Sphinx)
- Structure:
  - Combined user guide and API reference: a mix of written and generated documentation
  - Located together with source code
- Methodology:
  - Input also from source code and generated code
  - Automation: Pipeline deploys user guide to website
- Documentation should be well maintained and up to date

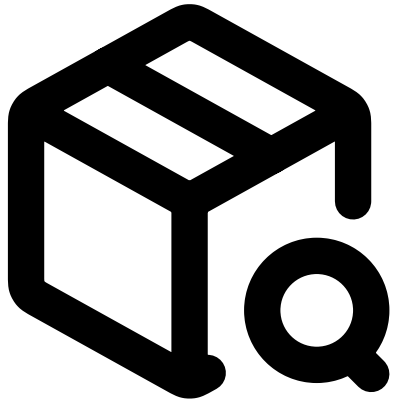


SINTEF
75 years

Common framework: Package management

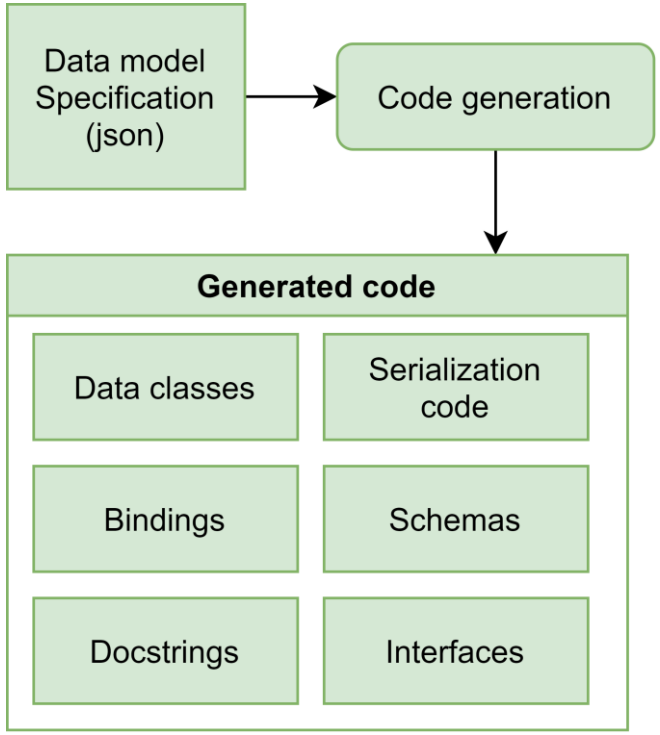


- Improved control of dependency artifacts
  - Internal dependencies and third-party dependenies
- Facilitates modularity in solutions
  - Libraries shared across projects
- Package managers:
  - **Conan**: Package management for C/C++
  - **Pip**: Package management for Python
- **Internal deployment**: package server
  - Used by developer environments and build runners
- **Supply chain security**: vulnerability scanning



Common framework: Code generation

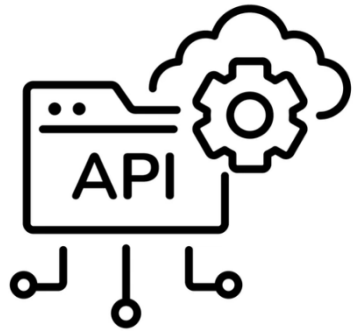
- Principle:
  - Single source of truth for data model (json file)
  - Generation of dependent source code
  - Utilize generated code as a library
- Benefits:
  - Reduces complexity of making changes
  - *Leads to greater consistency across code base*
- Used in ngLTM; other models may follow



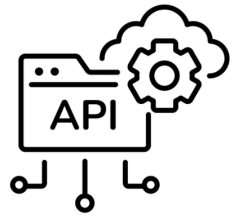
Common framework: **Libraries**

Shared code between projects:

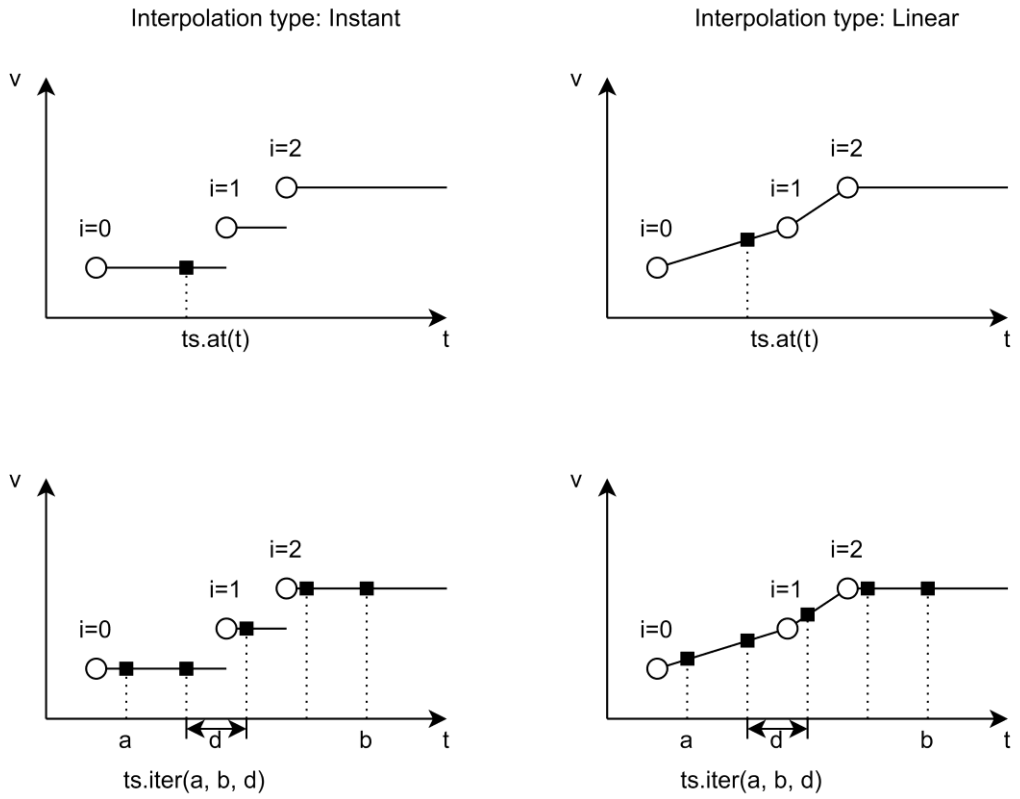
- From the ngLTM Project:
  - ngltm-timeseries C++ library
  - LTMIO C++ library
- Quality attributes of shared libraries:
  - Provides a generic API
  - Importable as build artifacts (Libraries and headers)
  - Deployed through package manager
  - Uses semantic versioning
  - High test coverage



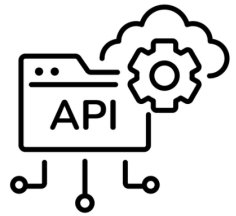
Common framework: ngltm-timeseries C++ library



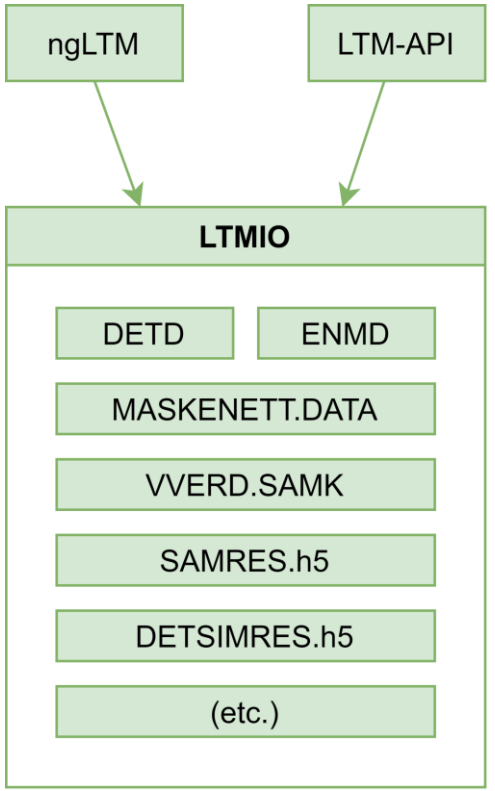
- Sampling, interpolation and iteration of irregular time series data
- Developed in ngLTM as a **stand-alone headers-only C++ library**
- Library used by ngLTM and LTM-API
- Out of scope for library:
  - Data distribution
  - Math



Common framework: LTMIO C++ library



- Common library for reading and writing LTM V10 files
- Developed in ngLTM project, then factored out and converted into a **stand-alone C++ library**
- Development and maintenance done by developers across projects within the LTM team
- Many files are supported; more are added as needed



SINTEF
75 years

Impact on
Prodrisk API and SHOP

- Prodrisk API
  - Prodrisk API has similar data modeling needs to ngLTM; will benefit from **code generation** in future.
- SHOP
  - SHOP has its own implementation details for objects and attributes, but which could also benefit from applying **code generation** principles in future.
  - Solutions for LP problem building in ngLTM will give insight for improvements to **memory allocation** in SHOP.

SINTEF

75 years

1950 – 2025
Technology for a better society

sintef.no/75