Open Simulation Software

Development and Application

TOM WARNKE[1], FRANK KRÜGER[2], AND ADELINDE M. UHRMACHER[1]

[1]Institute of Computer Science,
University of Rostock

[2]Institute of Communications Engineering,
University of Rostock

This talk in a nutshell

This talk shows

- how a continuous open pipeline from simulation software development via deployment to application by the end user (applying best practices and industry standards) can be implemented, and
- how persistent, reproducible, platform-independent simulation experiment artifacts can be provided.

Open science is trustworthy

- Availability and accessibility
  - The code for model and experiment is publicly available and readable.
- Software quality
  - The software's source code and development history is publicly available.
  - The software is regularly updated and bugs are fixed.
- Repeatability and distribution
  - Published results can be easily reproduced by repeating the simulation.
  - All software necessary to repeat a simulation is publicly available.
- Identifiers and connections
  - Relations between publications, models, experiments, data, etc. are captured

Trustworthiness of simulation research is a hot topic

2015: The ACM Task Force on Data, Software, and Reproducibility in Publication: Artifact Review and Badging in the ACM Digital Library (e.g., TOMACS, PADS)

Artifacts Evaluated acm Functional
Artifacts Evaluated acm Reusable
Artifacts Available acm
Results Replicated acm
Results Reproduced acm

2016: Uhrmacher, Brailsford, Liu, Rabe, Tolk: WSC Panel discussion on “Reproducible research in discrete event simulation – A must or rather a maybe?”

2017: Taylor et al.: WSC Paper on “Open Science: Approaches and Benefits for Modeling & Simulation”

2018: WSC introduces track on “Simulation Standards and Reproducibility”

Implementing Best Practices

The following slides show what “works for us” (the Modeling and Simulation Group at the Department of Computer science of the University of Rostock)

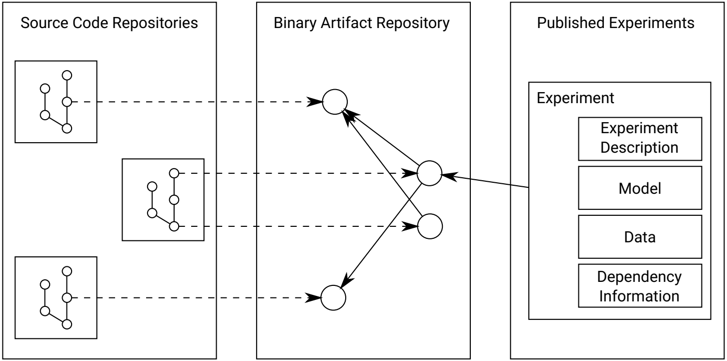
It combines best practices for

- Open Science
- Reproducibility
- Software Development

and exploits state-of-the-art technologies of the Java ecosystem.

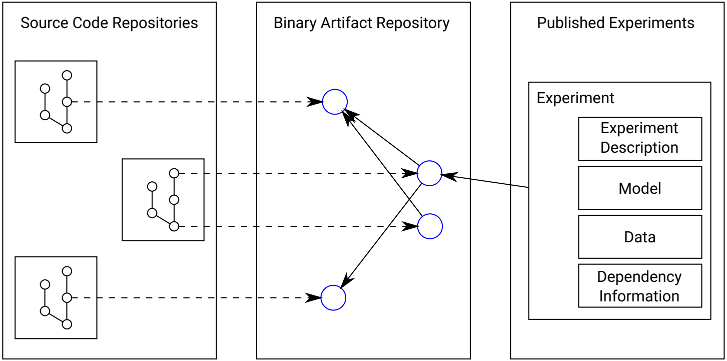
sessl.org

Overview


Source Code Repositories
Binary Artifact Repository
Published Experiments
Experiment
Experiment Description
Model
Data
Dependency Information

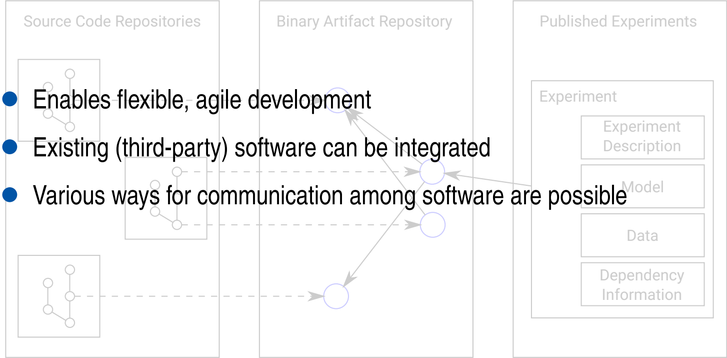
Separation of concerns

Loosely coupled small software projects



Separation of concerns

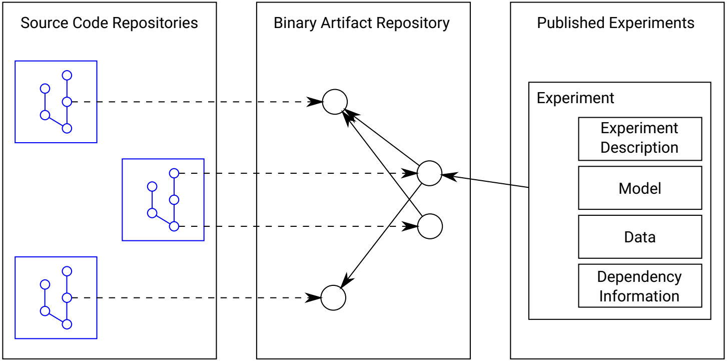
Loosely coupled small software projects



- Enables flexible, agile development
- Existing (third-party) software can be integrated
- Various ways for communication among software are possible

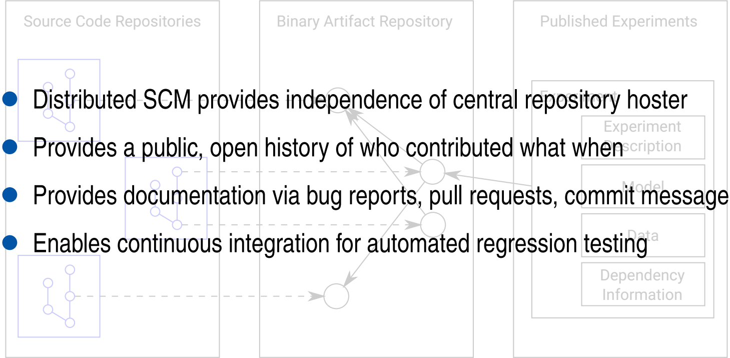
Source code management (SCM)

Public Git repositories with continuous integration



Source code management (SCM)

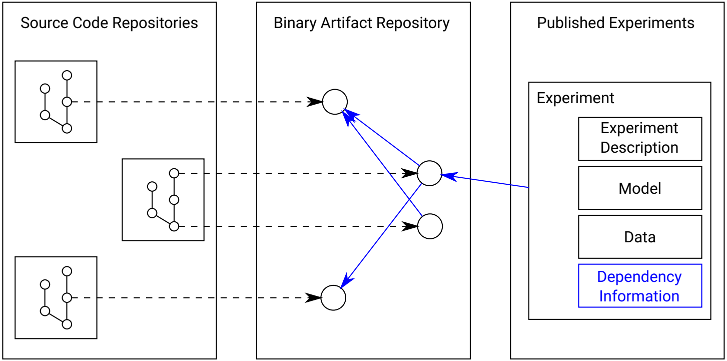
Public Git repositories with continuous integration



- Distributed SCM provides independence of central repository hoster
- Provides a public, open history of who contributed what when
- Provides documentation via bug reports, pull requests, commit messages
- Enables continuous integration for automated regression testing

Software management

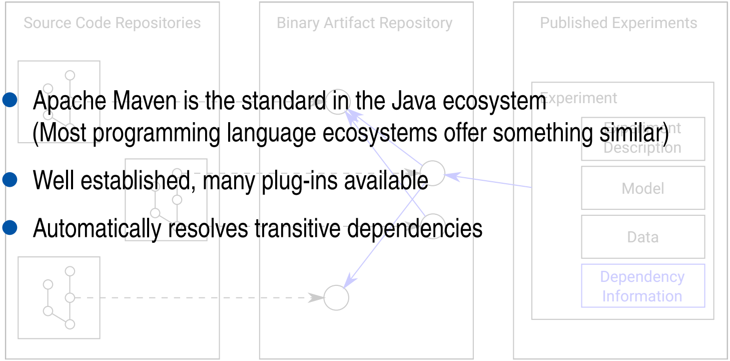
Build tools for project life cycle and dependency management



Software management

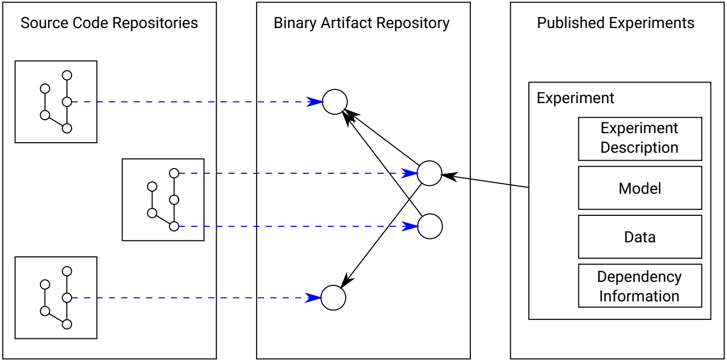
Build tools for project life cycle and dependency management

- Apache Maven is the standard in the Java ecosystem
(Most programming language ecosystems offer something similar)
- Well established, many plug-ins available
- Automatically resolves transitive dependencies



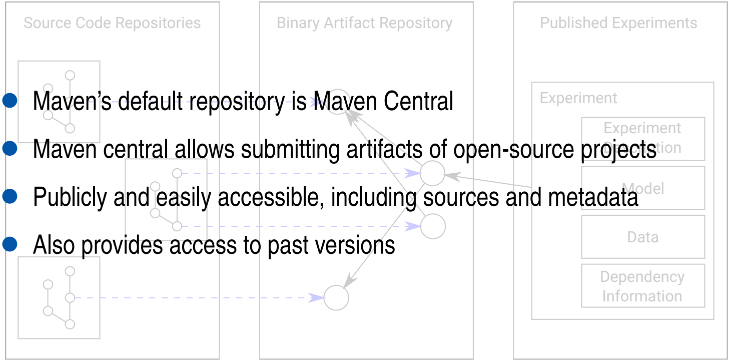
Deployment

Publish binary artifacts to standard public repositories



Deployment

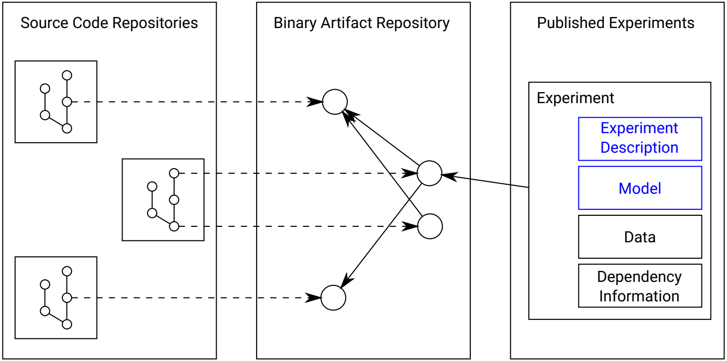
Publish binary artifacts to standard public repositories



- Maven’s default repository is Maven Central
- Maven central allows submitting artifacts of open-source projects
- Publicly and easily accessible, including sources and metadata
- Also provides access to past versions

Domain-specific languages (DSLs)

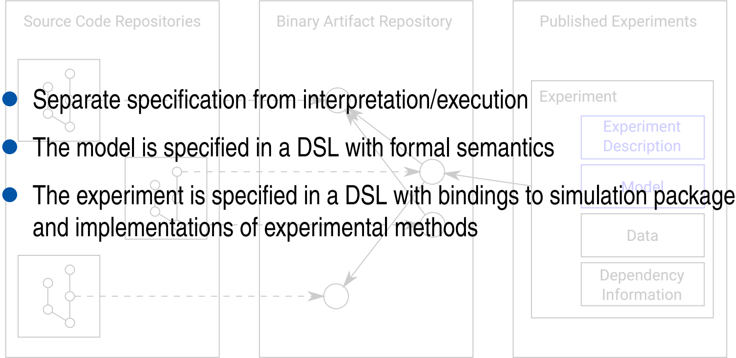
Readable, unambiguous descriptions



Domain-specific languages (DSLs)

Readable, unambiguous descriptions

- Separate specification from interpretation/execution
- The model is specified in a DSL with formal semantics
- The experiment is specified in a DSL with bindings to simulation packages and implementations of experimental methods



Source Code Repositories

Binary Artifact Repository

Published Experiments

Experiment

Experiment Description

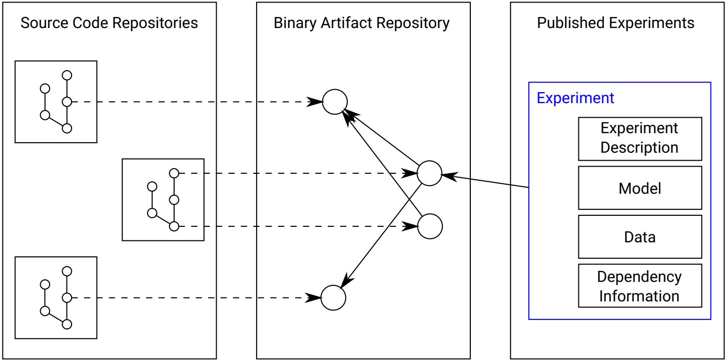
Model

Data

Dependency Information

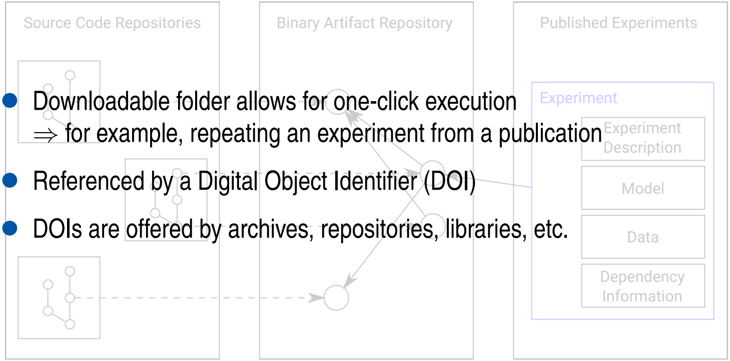
Linkable experiment artifacts

Executable experiment persisted and identifiable



Linkable experiment artifacts

Executable experiment persisted and identifiable



- Downloadable folder allows for one-click execution
  ⇒ for example, repeating an experiment from a publication
- Referenced by a Digital Object Identifier (DOI)
- DOIs are offered by archives, repositories, libraries, etc.

Alternatives and possible future developments

- Docker containers
- Innovative package managers such as Nix
- Databases for experiments similar to the ones for models
  - Standards?
  - Formalization?
- Recording provenance information

Take-home messages

How to facilitate open trustworthy simulation research

- Make your code open-source. Also think about licensing.
- Employ CI for QA. Travis CI is free for open-source projects on GitHub.
- Exploit the ecosystem of your programming language. It probably offers some way to distribute software packages.
- Make running experiments as easy as possible. It should be possible on any major OS without root/admin privileges.
- Provide old versions of your software. Beware of breaking changes that make running old experiments impossible.