Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

Version Control Architecture

Assembly Process Architecture

ARCHITECTURE FUNDAMENTALS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Development: Build Standard Process — DEV

DS — (Source) Distribution Site

source:upload
source:download

A — Archived — B

source:pack — pack
source:unpack — unpack

source:track — track
source:checkout — checkout
source:checkin — checkin
source:lint

VCS — Version Control System

F — Fresh
B — BootStrapped

stage0[:build] bootstrap generate
stage0:clean bootstrap:clean realclean
stage0:lint bootstrap:lint

stage1[:build] setup configure
stage1:clean setup:clean distclean
stage1:lint setup:lint

S — Setup

Roles legend:
- Vendor
- Packager (Source)
- Packager (Binary)
- Deployer
- Operator

DEVELOPER LOOP
1. edit
2. compile
3. start
4. stop

1 PREPARATION
Subversion, Git, Mercurial, Maven, Gradle, NPM, YARN, UPD, Autoconf, Automake, CMake, OpenPKG RPM

AR — Artifact Repository

artifact:download
artifact:upload

A — B

Software Developer
Release Engineer
Continuous Integration System
Continuous Deployment System

stage2:test compile:test test
stage2[:build] compile
stage2:clean compile:clean clean [1]
stage2:lint compile:lint lint

C — Compiled

stage3:test link:test test
stage3[:build] link build
stage3:clean link:clean
stage3:lint link:lint

L — Linked

stage4:test bundle:test
stage4[:build] bundle all
stage4:clean bundle:clean
stage4:lint bundle:lint

B — Bundled

source editing

SC1: Developer Short-Circuit Transition
SC2: Developer Short-Circuit Transition
SC3: Developer Short-Circuit Transition

2 BUILDING
Make, Maven, Gradle, Ant/Ivy, NPM, YARN, Docker Build, OpenPKG RPM

---

stage5:test package:test test
stage5[:build] package:pack package
stage5:clean package:clean
stage5:lint package:lint

P — Packaged

package:upload
package:meta-write

SC4: Developer Short-Circuit Transition

DS — (Binary) Distribution Site

A — B

3 DISTRIBUTION
NSIS, InnoSetup, Tar, InfoZip, NPM, Maven, Gradle, Docker, OpenPKG Build

---

BUILD PATTERNS

mode	lossless	lossy
generation	(example: editing from scratch, making random data)	✗
duplication	(example: downloading, uploading)	✗
fragmentation	(example: unpacking archive)	(example: object extraction)
transformation	(example: format changing, recoding)	(example: source compilation, parsing)
integration	(example: packing archive)	(example: linking objects)
elimination	✗	(example: cleaning up)

operation

Operations: Deployment Standard Process — OPS

runtime:backup backup
runtime:restore restore
runtime:test test
runtime:start start
runtime:status status
runtime:reconfigure reconfigure
runtime:reload reload
runtime:stop stop
runtime:scrub scrub

R — Running

GLOBAL RULES
- **ROUND TRIP:** every forward-only activity should have a corresponding backward activity
- **CONSISTENCY:** directly triggering an activity causes intermediate activities (between old and new state) to be implicitly triggered first
- **DEVELOPMENT:** there are intentional short-circuit transitions between states
- **ACTIVITY NAMING:** each activity has at least an official name and zero or more calling aliases
- **AUTOMATION:** each activity has to be automated inside a command-line driven tool (and is just triggered from a control head tool)

6 RUNTIME
OpenRC, SystemD, Init, SupervisorD, OpenPKG RC, Docker

---

install:upgrade upgrade
install:update update

C/L/B — Compiled/Linked

install:configure configure
install:install install
install:uninstall uninstall

I — Installed

install:migrate migrate
install:validate validate
install:repair repair

B — Bundled

5 INSTALLATION
APT, dpkg, RPM, OpenPKG RPM, Docker

---

package:meta-read

package:unpack
package:download

P — Packaged

4 RESOLUTION
APT, dpkg, RPM, OpenPKG Build, Docker

Legend:
- X (blue) External State/Resource
- X (gray) Process State
- → State Transition
- ---→ State Transition (short-circuit)
- → State Transition (external)
- XXX (white) Process Activity (semi-automated or automated)
- XXX (gray) Process Activity (manually or semi-automated)

Environments & Quality Assurance

Artifact Flow

DEV   INT   TST   SBX

Version Control System (Development)

Artifact Repository (Development)

Artifact Repository (Operations)

STG   DRY   PRD   FOV

Development Environments

DEV — Development
(Potentially destructive) Environment for the (separated) development of all artifacts of the system, typically located on each developer's respective system.

INT — Integration
(Potentially destructive) Environment for integrating all artifacts of the system, usually located centrally to the developers and driven automatically.

TST — Test
Environment for testing the system as a whole in a production-resembling context.

SBX — Sandbox
Destructive environment mirroring the Test environment for demonstration and experimentation purposes, including trainings and showcases.

Operations Environments

DRY — Dry-Run
Destructive environment mirroring the Staging environment for experimentation purposes, including operating system and major dependency upgrades.

STG — Staging
Environment for staging the system as a whole in a production-equal context in order to deploy it to the Production environment subsequently.

FOV — Failover
Destructive environment mirroring the Production environment for failover situation, including disaster recovery situations.

PRD — Production
Environment for running the system as a whole in production.

Environment Mirroring

Replicated Environments

Original Environments

Testing Approaches

Testing Targets:

UT — Unit Testing
Whitebox testing the inner details of individual units (components) against their functional specification.
Specialization: Regression Testing

IT — Integration Testing
Whitebox testing the outer interplay of individual units (components) against the technical design of the solution.
Specialization: Smoke Testing

ST — System Testing
Blackbox testing the system as a whole against the functional and non-functional requirements of the solution.
Specialization: Load Testing

UAT — (User) Acceptance Testing
Blackbox testing and formal approval of the system as a whole against the end-to-end functionality and user experience requirements.

OT — Operation Testing
Blackbox testing the system as a whole against the availability and proper operation of the intended services.
Specialization: Service Monitoring

Quality Promise, Bugfix Effort

DevOps Toolchain

TECHNISCHE UNIVERSITÄT MÜNCHEN

Development Environment

Developer

ED

SC DS BS TS

WC

SC DS BS TS

IC

BX

TC DS TS IC

DC

TC DS TS IC

DX

TC

RT

Integration Environment

SC DS BS TS

VC

CI

SC DS BS TS

IC

BX

TC DS TS IC

AR

TC DS TS

CD

DX

TC

RT

Developers

Test Environment

Actor-Store Pattern

Actors

A

Stores

S   S

CD

TC DS TS

DX

TC

RT

Testers

Staging Environment

DevOps Pipeline Pattern

Development   Operations

Actors

CI   CD

Stores

VC   AR   RT

TC DS TS

CD

DX

TC

RT

Production Environment

Administrator

IA

TC DS

DX

TC

RT

Users

Rules

Rule 1: Actors are linked either by Store-connected artifact flows or by plain triggers.
Rule 2: The configuration of all Actors and Stores has to be kept as small as possible by just performing orchestration and retrieving the actual commands via external scripts.
Rule 3: On each Version Control (VC) commit, the application should be automatically redeployed as an updated version on the Run-Times (RT).
Rule 4: Development and Operations can be split via two synchronised Artifact Repositories (AR), because of network topology constraints, or act on a single common one.
Rule 5: Deployment on Staging can be triggered automatically or manually.

Actors

ED: Editor / IDE
CI: Continuous Integration
BX: Build Execution
CD: Continuous Deployment
DX: Deployment Execution
IA: Infrastructure Automation

Stores

WC: Working Copy
VC: Version Control
DC: Distribution Copy
AR: Artifact Repository
RT: Run-Time

Flows

→ Artifact Flow
⇢ Event Flow
⇢ Control Flow

Artifacts

SC: Source Component
IC: Interm. Component
TC: Target Component
BS: Build Script
DS: Deployment Script
TS: Test Script

Development   Operations

Software Refinement Process

ARCHITECTURE FUNDAMENTALS

TECHNISCHE UNIVERSITÄT MÜNCHEN

DT Develop. Tools
BE Build Env.
PS Package Spec.
PF Pack. Framework
OS Operat. System
HW Hardware
OE Operation Env.

SN Specification
C Coding
SW Software
B Building
SC Soft. Component
P Packaging
SP Soft. Package
A Assembling
SS Software Stack
D Deploying
SA Soft. Appliance
I Installing
HA Hard. Appliance
O Operating
SV Service

Generalization

Specialization

Software Refinement Value Creation Chain

Persons/Roles

- Software Engineer
- Software Builder
- Software Packager
- Solution Engineer
- System Integrator
- System Engineer
- System Administrator
- DevOps Engineer

Companies

- Full-Service Provider
- Independent Software Vendor (ISV)
- Operating System Vendor
- Software Distribution Vendor
- System Integrator
- Value Added Reseller (VAR)
- Managed Hosting Provider
- Datacenter Provider

Software Release Management

Evolution Stage — What?

WF — Wireframe
Distraction-free low-fidelity illustration of the solution and its base features, displaying its pure structure and core functionality only.

WS — Walking Skeleton
Realization of all technical, fundamental aspects of the solution, ensuring the domain specific aspects can be realized later on top of it without problems.

PT — Prototype
High-fidelity mostly interactive sample, mockup, model or simulation of the solution and its base features, showcasing its structure and functionality.

MVP — Minimum-Viable Product
Early version of solution with just enough functionality to enable full turn of Build-Measure-Learn loop with minimal amount of effort and time.

PoC — Proof of Concept
Pure realization of most-risky aspects of the solution, proofing their feasibilities. Might still be based on a different technology than WS, MVP and FP.

FP — Full Product
Final version of the solution with all intended functionality and targeting the mainstream market.

arbitrary technology ← → *target technology*

Version Number — When?

M — Major Version
Major Version of solution. Usually bumped on major technical or domain-specific changes only. A bump resets the Minor Version and the Revision, too.

N — Minor Version
Minor Version of the solution within the Major Version. Usually bumped on new features. A bump resets the Revision, too.

R — Revision
The Revision of the Release Phase within Major and Minor Version. Bumped for every A/B/C/R Release Phase.

Release Phase (p) — When?

A — Alpha
Early version of the solution with incomplete and unstable functionalities to get feedback on product. Usually tagged as "M.NaR" (R > 0).

C — Candidate
Mature version of solution with complete and stable functionalities to catch last-minute problems. Usually tagged as "M.NrcR" (R > 0) around RTM.

B — Beta
Early version of the solution with complete but still unstable functionalities to stabilise product. Usually tagged as "M.NbR" (R > 0).

R — Release
Release version of the solution with complete and stable functionalities, available for production use. Usually tagged as "M.N.R" (R >= 0).

Points-In-Time (PiT) — When?

DEV — Development
Arbitrary permanent points-in-time during development. This is the default tag for the source code. Intended for no availability releases.

SNP — Snapshot
Distinct temporary point-in-time for a release of the current version without a version increase. Intended for limited availability releases.

REL — Release
Distinct temporary point-in-time for a release of the current version with a version increase. Intended for early and general availability releases.

Product Edition — Which?

CE — Community Edition
Edition of the solution for the Open Source Community. Contains just the base functionality and has limited volunteering support.

STD — Standard Edition
Edition of the solution with just the standard functionality and regular support.

EE — Enterprise Edition
Edition of the solution for the Enterprise market. Contains the base and additional functionality and has full commercial support.

PRO — Professional Edition
Edition of the solution with both the standard and extra functionalities and extended support.

Availability Scope (S) — Who?

XA — No Availability
No public availability of solution at all. The scope for all Development and sometimes Snapshot point-in-times.

EA — Early Availability
Early public availability of solution for early market. Usually for Beta or Release Candidate levels or for Release and initial Release Update levels.

LA — Limited Availability
Limited public availability of solution. Usually for releases after the End-of-Life-Announcement (EOLA) or for releases with specific customer features.

GA — General Availability
Late public availability of solution for mainstream market. Usually for Release and sometimes just for Release Update levels.

Distribution Channel — Where?

BLEED — Bleed Channel
Distribution channel for all daily snapshots ("YYYY.MM.DD") with experimental features turned on. Intended for testing purposes only.

STABLE — Stable Channel
Distribution channel for all quarterly releases ("YYYY.QN") with experimental features turned off. Intended for fast mainstream market and production use.

EDGE — Edge Channel
Distribution channel for all monthly releases ("YYYY.MM") with experimental features turned on. Intended for early market or testing purposes.

SOLID — Solid Channel
Distribution channel for all (half-)year releases ("YYYY[.N]") with experimental features turned off. Intended for slow mainstream market and production use.

Distribution Channel Artifacts

SOLID · STABLE · EDGE · BLEED

1 day · 4 weeks · 3 months

Versioning Scheme (stdver.org)

M . N p R [. D] [+ H] [- S]

Examples:
1.2a3.20230101+F42A–XA
1.2b3–EA
1.2.3–GA
1.2.3

Product Life-Cycle

Release to Manufacturing (RTM) · End-of-Life Announcement (EOLA) · Last-Order-Date (LOD) · End-of-Life (EOL)

t

XA · LA · EA · GA