

Follow-up: Towards a Common GSI/FAIR Software Development Guideline

- “The Only Unbreakable Law”: <https://indico.gsi.de/event/16522/>
- “Where Does Bad Code Come From?": <https://indico.gsi.de/event/16533/>
- Initial Guideline Proposal: <https://indico.gsi.de/event/18737/>

Ralph J. Steinhagen

C++ User-Group Meeting

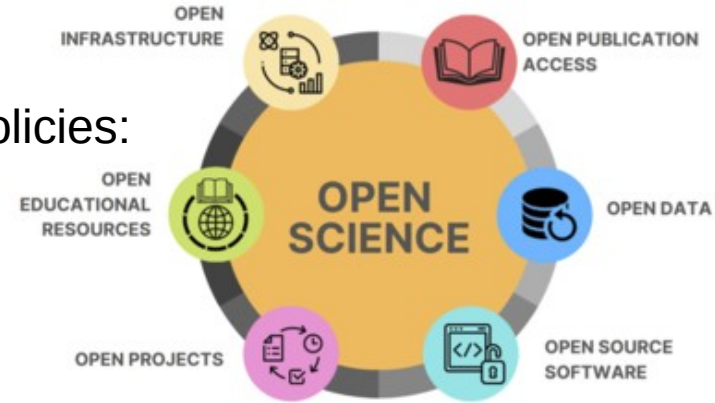
February 7th, 2024



Why a Software Development Guideline?

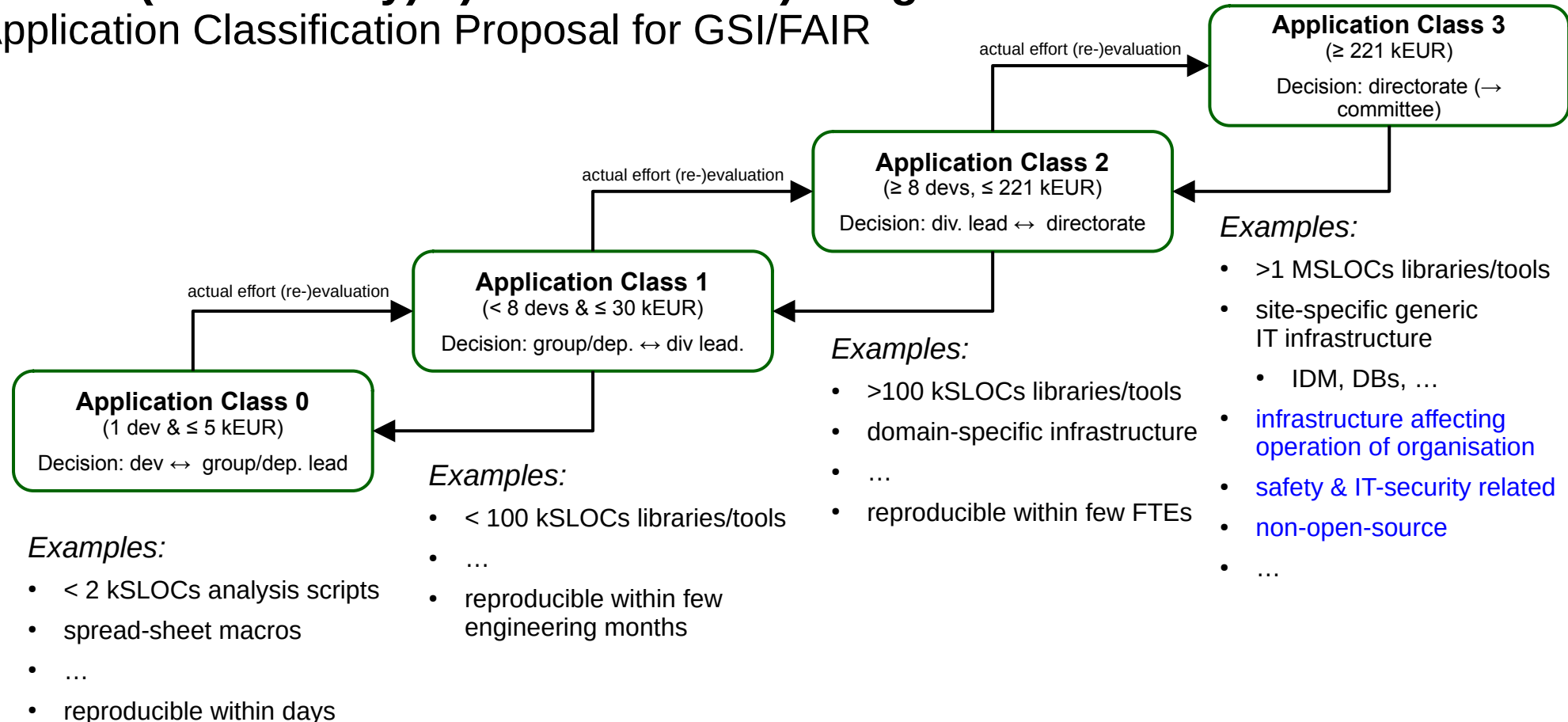
not fundamentally 'new' – existing GSI/FAIR Open Science policies:

- since 2021: [Open Source Guidelines](#)
 - GPLv3 → LGPLv3 → Apache 2.0 & CERN OHW
especially for public-public & public-private collaborations
 - proprietary licenses/development
require [documented](#) evaluation of total-cost-of-ownership & [risk](#) for the organisation
- since 2023: Open-Science WG Initiative
[GSI/FAIR supporting 'Public Money? Public Code!' Campaign](#)
 - new directorate policy:
favour open-source and document risks related to
developments and procurements of new software systems
- [2023/2024: Open-Science WG charged by directorate to draft RSE guidelines for GSI/FAIR](#)
 - refine definition for Open-Hardware/Software policies
 - update licenses (N.B. new AI-regulations and CE certification)



HowTo (realistically) a) assess and b) mitigate Risks related to RSE

Application Classification Proposal for GSI/FAIR



greater scrutiny with greater risks

organisation

employee



HowTo (realistically) a) assess and b) mitigate Risks related to RSE

Proposed Measures & Metrics

Application Class 1

(decision-level: research group/department)

1. Documentation:

- README.md
- [CORE DEVELOPMENT GUIDELINE.md](#)
quantifiable metrics, lean and clean code management inspired by the Toyota Production System

2. Code Quality and Automation:

- recommended: public repository i.e. <https://github.com/>, or <https://git.gsi.de/>
- CI pipeline → metric automation
- Source Lines of Code (kSLOC) & COCOMO-II^{1,2} estimates
- functional unit and integration tests with a defined minimum code coverage percentage.
- application-related (micro-) benchmarks
(↔ mitigates premature optimisation risk)
- [mandatory code review process \(soft QA\)](#)

3. Coding Standards:

- Language Core Guide-Lines
- team-specific specialisations

4. Code Maintenance and Style:

- [Documentation of dependency hierarchy and – where known – downstream dependents.](#)
- Warning & Static-Code-Analysis reporting via e.g. Sonar, Snyk, ... (-Werror)
- [CORE NAMING GUIDELINE.md](#) & [CORE FORMATTING GUIDELINE.md](#)
(e.g. via LLVM tooling)
- [Boy-Scout-Rule](#)

5. Project Management:

- [Bus-Factor ≥ 1-2 ??](#)
- *recommended: Kanban methodology & accounting*

Application Class 2

(decision-level: collaboration/division)

1. Inherits Class 1 requirements

2. Project Lifecycle:

- define lifetime and maintenance expectations
(e.g., 3 years, 5 years, indefinite)

3. Open Source and Standards:

- F(L)OSS & F.A.I.R.
- LICENSE: either GPLv3, LGPLv3 (ext. or ind. co-use), or Apache 2.0
- Public repository i.e. <https://github.com/>, or <https://git.gsi.de/>
- *recommended: regular security and GDPR audits*

4. Community Engagement:

- [CODE OF CONDUCT.md](#)
- [CONTRIBUTING.md](#)

5. Continuous Delivery: CI & CD pipeline

6. Project Management Enhancements:

- [Bus-Factor ≥ 3 \(global project\) \(N.B. definition for sub-components!\)](#)
- Implement agile Kanban methodology: backlogs, epics, issues, story points (estimation vs. actual), development velocity, and burn-down charts.
- [track personnel and resource spending](#)
- *recommended: evaluate total-cost-of-ownership*

Application Class 3

(decision-level: organisation, directorate → committee)

1. Inherits Class 1 & 2 Requirements

2. Enhanced Project Lifecycle and Governance:

- [critically define project lifetime and maintenance plans, ensuring alignment with organisational strategies and objectives.](#)
- [evaluate total-cost-of-ownership to inform decision-making processes.](#)

3. Risk Management and Compliance:

- [regular security audits and privacy impact assessments, especially for sensitive and critical infrastructure](#)

4. Transparency and Collaboration:

- [promote a culture of transparency by openly sharing source code, development processes, and project governance, fostering community contributions](#)
- deploy and utilise open infrastructure that supports open collaboration

5. Avoiding Vendor Lock-in:

- prioritise use of open standards and open-source software to maintain organizational control over technological infrastructure and ensure adaptability
- integrate vendor lock-in avoidance strategies into project planning and procurement, favouring open-source alternatives.
- mandatory documentation & risk-analysis for proprietary and closed-source solutions

6. Sustainability and Efficiency:

- highlight open-source software's role in fostering sustainable and efficient RSE practices, such as reusing existing solutions and contributing to community projects
- encourage contributions to the sustainability of the open-source ecosystem, including upstream project improvements and community engagement

7. Attracting Talent:

- utilise the commitment to FLOSS as a draw for talented software engineers who prioritise transparency and public contribution
- prominently feature successful open-source projects and contributions in recruitment and outreach initiatives

8. Scalability and Performance:

- implement scalability and performance benchmarks to ensure projects can accommodate organisational growth and evolving needs.

Example: Kanban Methodology

Pool of Ideas	Feature Preparation		Feature Selected	User Story Identified	User Story Preparation		User Story Development		Feature Acceptance		Deployment	Delivered
	3 - 10		2 - 5	30	15		15		8		5	
Epic 431	In Progress	Ready			In Progress	Ready	In Progress	Ready (Done)	In Progress	Ready		Epic 294
Epic 478	Epic 444	Epic 662	Epic 602			Story 602-02 Story 602-03	Story 602-06 Story 602-04	Story 602-05 Story 602-01	Epic 401	Epic 609	Epic 694	Epic 386
Epic 562	Epic 589		Epic 302	Story 302-03 Story 302-02	Story 302-01 Story 302-06	Story 302-07 Story 302-08	Story 302-09	Story 302-05 Story 302-04	Epic 468	Epic 577	Epic 276	Epic 419
Epic 439	Epic 651		Epic 335	Story 335-09 Story 335-08	Story 335-10 Story 335-01 Story 335-03	Story 335-04 Story 335-05 Story 335-02	Story 335-06 Story 335-07		Epic 362		Epic 339	Epic 388
Epic 329			Epic 512	Story 512-04 Story 512-05	Story 512-07 Story 512-06 Story 512-03	Story 512-02	Story 512-01				Epic 521	Epic 287
Epic 287											Epic 582	Epic 274
Epic 606	Discarded											
	Epic 511	Epic 213										
	Epic 221											

Policy

Business case showing value, cost of delay, size estimate and design outline.

Policy

Selection at Replenishment meeting chaired by Product Director.

Policy

Small, well-understood, testable, agreed with PD & Team

Policy


As per "Definition of Done" (see...)

Policy

Risk assessed per Continuous Deployment policy (see...)

Example: PSP 2.14.17 Day-to-Day Project View

<https://github.com/orgs/fair-acc/projects/5/views/1>

 Product Solutions Open Source Pricing

Search or jump to... [7] Sign in Sign up

Digitizer Reimplementation

Backlog Call#4 Details Call#1

Filter by keyword or by field Discard

Ideas (∞) 3

opendigitizer #2

Test

optional

graph-prototype #82

scheduler: define API between graph and scheduler

opencmw-cpp #300

37 dns with persistence

Backlog 27

graph-prototype #81

graph: refactor node::work() function

opendigitizer #90

[0pt] See what the problem is with memory relocating ports after connection

opendigitizer #23

[8pt] UI: A basic 3D plot example for Implot (mountain range or 3D surface plot) for technology evaluation

opencmw-cpp #202

[2pt] Review settings management for OpenCMW workers

opencmw-cpp #203

[3pt] Implement failure and error handling in OpenCMW

opendigitizer #12

[5pt] UI: User can modify, regroup, split signal into sets or single views and perform simple grid/box layout adjustments

opendigitizer #14

[5pt] UI: Zoom and panning should work with mouse and touch where available

opendigitizer #18

[2pt] UI: Feature and parameter extraction (minimum, maximum, location of maximum, pulse width...)

opendigitizer #17

[2pt] UI: Support for scalar values to be plotted as history (i.e. actual value versus wall-clock time)

Selected (3) 7

opendigitizer #42

EPIC: Refactoring low-level GR 4.0 API

opencmw-cpp #214

Fix and document mapping between Rest Backend URIs and majordomo service name and topic

opendigitizer #33

[3pt] GR Flowgraph: Modernize the codebase

opendigitizer #34

[3pt] GR Flowgraph: Hierarchical (hier) block support

graph-prototype #69

GR4: implement selector block/node (switch matrix like run-time plumbing between input->output ports)

opendigitizer #71

optional (at this stage): graph/block profiling (enable/disable at compile-time) -- proposal: re-use chrome tracing format's & UI infrastructure (e.g. chrome://tracing/ and Peretto)

opencmw-cpp #183

[5pt] finish CmwLight/rda3 client/protocol

enhancement

In progress 6

opendigitizer #69

GR4.0: FFT block implementation

opendigitizer #37

[5pt] Non-distributed DSNS with persistence

opencmw-cpp #194

[3pt] Review, improve and formalize the URI protocol

opencmw-cpp #191

[2pt] Event store: Evaluate the current action settings and extend if needed, transfer from OpenCMW to GR

opendigitizer #15

[5pt] UI: Storing, clearing and reloading the flow-graph and the UI configuration

opendigitizer #91

[0pt] Investigate all the node instances we use in tests and extract into a library

Finished Implementation (2) 0

QA-Accepted/Merged (∞) 49

opendigitizer #95

[0pt,5.5pt] graph-prototype: Add support for loading and storing graphs in YAML files

#95 #101

opendigitizer #36

[3pt,7pt] UI and GR Flowgraph: Modify block properties during runtime

#92

opencmw-cpp #197

[3pt,8pt] Integrate GNU Radio sinks with the OpenCMW Disruptor

#80

graph-prototype #76

graph: multithreaded schedulers

#78

opendigitizer #73

[5pt,9pt] GR Flowgraph: Block registry, runtime data, bootstrapping the flow-graph from static reflection data

#83 #88

opendigitizer #11

[5pt, 6pt] UI: Simple configurable layout of views - grid layout, horizontal and vertical box layout, spanning grid view

#88

graph-prototype #75

GR 4.0: node and -field annotation interface

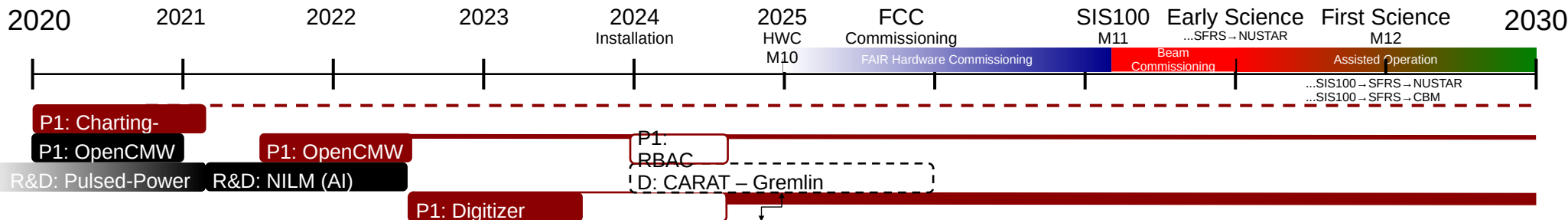
#77

opendigitizer #10

[2pt, 1pt] UI: Layout or view detaching to a separate window

Example: PSP 2.14.17 Project Timeline

CSG-based Prioritisation P1→P4 (status as of 2023-07-01)

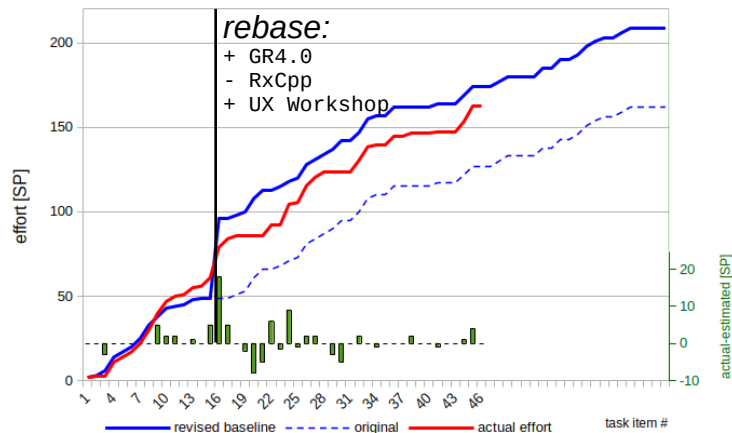


Available Resources:

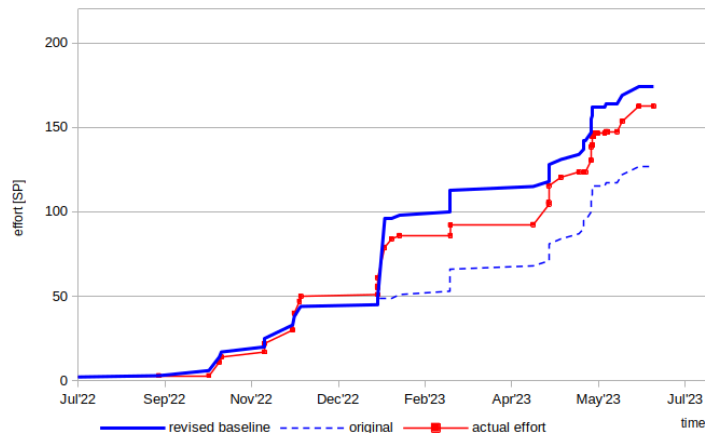
- GSI: 2 FTE (dist. over 3 people)
- KDAB: 1½-2 FTE (dist. over 5+1 developer)
(N.B. PSP liquidities/funding until 2026)

SYS responsible for 4 ACC Core Technologies

(↔ internal/external user support, liaison, contracts, ...)



<https://github.com/orgs/fair-acc/projects>



<https://github.com/orgs/fair-acc/projects/5>

unscheduled CSG items:

- P3: Machine-Experiment Interface
- P3: Status & Overview SIS18
- P3: Status & Overview SIS100
- P3: Beam-Quality-Monitoring
- P3: Slow-Extraction Optimisation I
- P3: SIS100 RF Bunch Gymnastics
- P3: Dig. Set-Actual Comparison
- P4: el. Pulsed-Load Monitoring
- P4: web-status/fixed-displays
- P4: multi-parameter-beam
- P4: Q/Q' control
- P4: Multi-Turn-Injection Control
- P4: Slow-Extraction Optimisation II
- P4: Optics Correction
- P4: Collimator/Cleaning Set-Up
- P4: Final-Focus Control
- P4: L/T Emittance Control
- P4: Post-Mortem Analysis
- P4: ...