

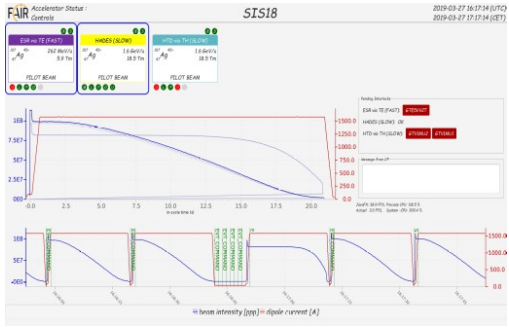
Open-Source Software Development with Industry

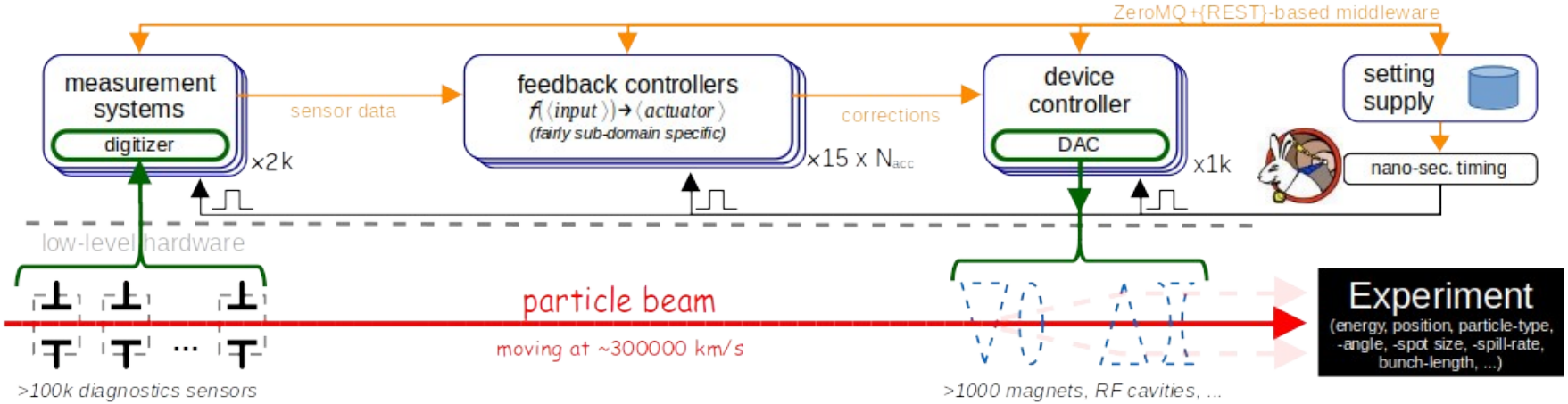
Alexander Krimm, SIS100/SIS18
1st Workshop on Open Science @ GSI/FAIR 20.10.2023

Finland France Germany India Poland Romania Russia Slovenia Sweden UK

GSI HELMHOLTZ ASSOCIATION FAIR

Who are we?

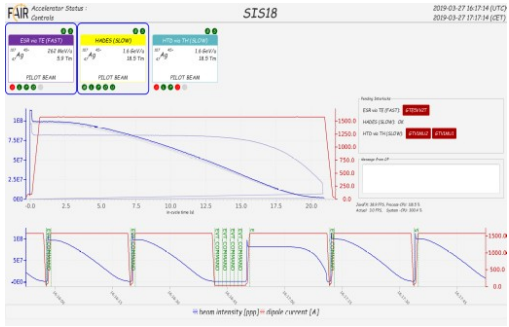

SIS18

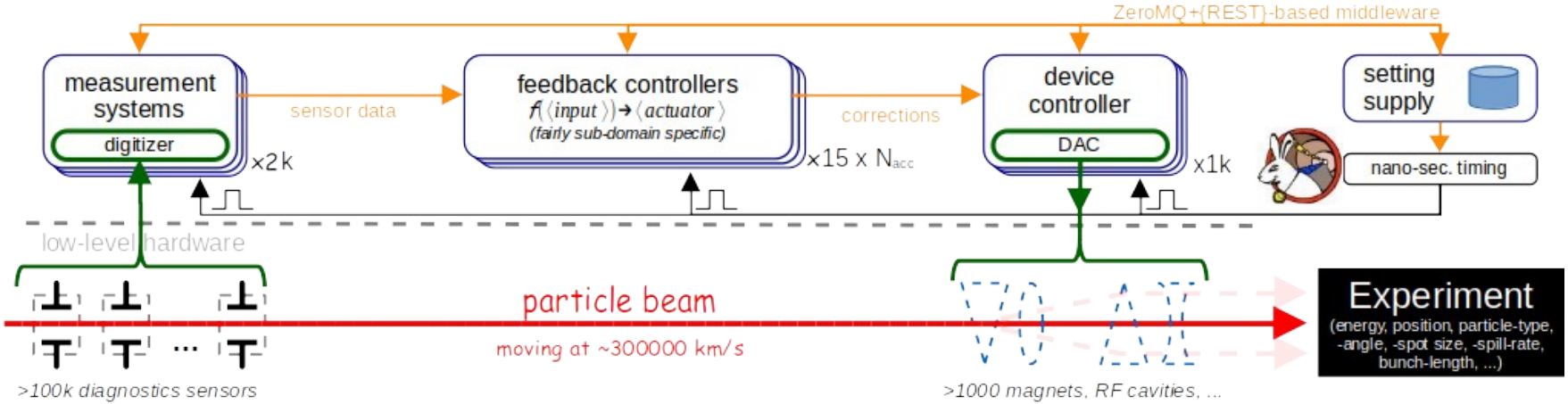

ZeroMQ+{REST}-based middleware
measurement systems
digitizer
×2k
sensor data
feedback controllers
f(⟨input⟩)→⟨actuator⟩
(fairly sub-domain specific)
×15 x N_acc
corrections
device controller
DAC
x1k
setting supply
nano-sec. timing
low-level hardware
particle beam
moving at ~300000 km/s
>100k diagnostics sensors
>1000 magnets, RF cavities, ...
Experiment
(energy, position, particle-type, -angle, -spot size, -spill-rate, bunch-length, ...)

Who are we?

small team in SIS100/SIS18 + help from CIT and ACO working on system integration, beam-based diagnostic and feedback systems for FAIR

- 100k signals from various data-acquisition sources & 100s of systems
- 20+ core services per machine for commissioning, first-line diagnostics and operation + co-use by experiments



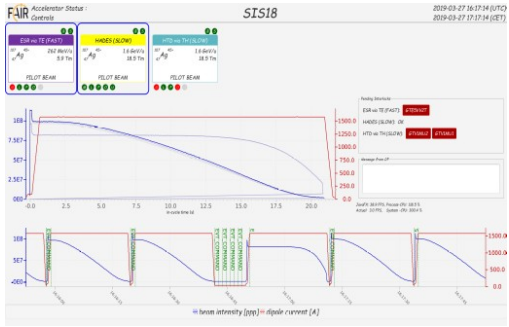


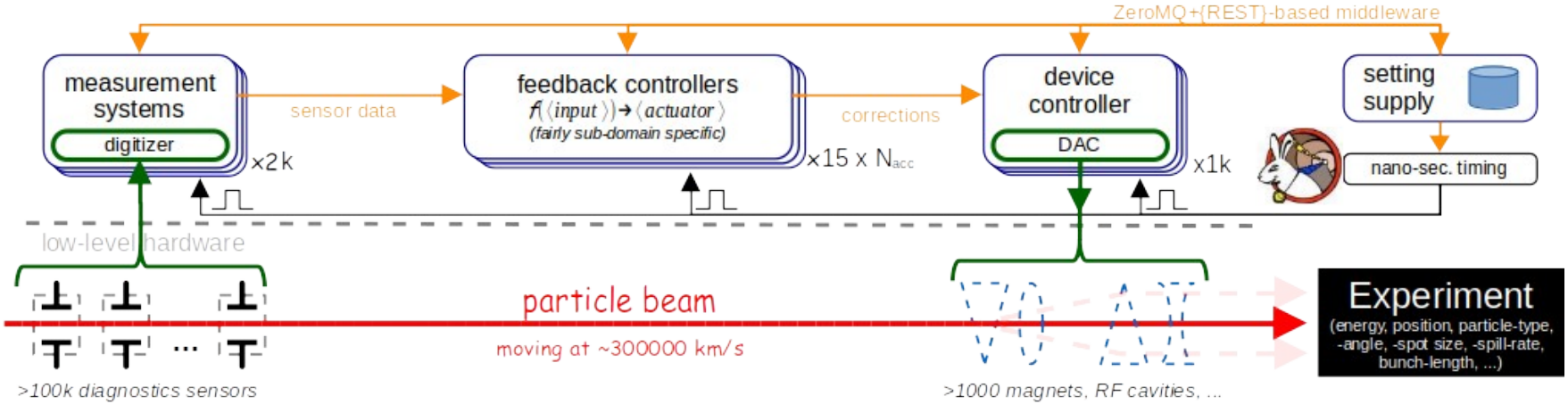
Who are we?

small team in SIS100/SIS18 + help from CIT and ACO working on system integration, beam-based diagnostic and feedback systems for FAIR

- 100k signals from various data-acquisition sources & 100s of systems
- 20+ core services per machine for commissioning, first-line diagnostics and operation + co-use by experiments
- **60+ person-years of scheduled work → only possible with external partners**

=> huge demand for **sustainable** software development





Motivation

Motivation

Why have external Partners at all?

- **lack of in-house resources:** freeing in-house team to focus on FAIR-accelerator-specific tasks.

Motivation

Why have external Partners at all?

- **lack of in-house resources:** freeing in-house team to focus on FAIR-accelerator-specific tasks.

1) Mitigating Risks → ‘agile’ and ‘lean’ Framework Contract
   - general FAIR contract designed for established technologies but unsuitable for SW development
2) Software Quality and Efficiency mirrors Communication Structure (→ M. Conway’s Law)
   - **collaborations must be based on trust + open and transparent communication**
3) Public Documentation of Technologies & Outreach
   - fostering seamless collaboration & attract a diverse talent pool
   - break free from vendor lock-in, circumvent single service-provider problem

Motivation

Why have external Partners at all?

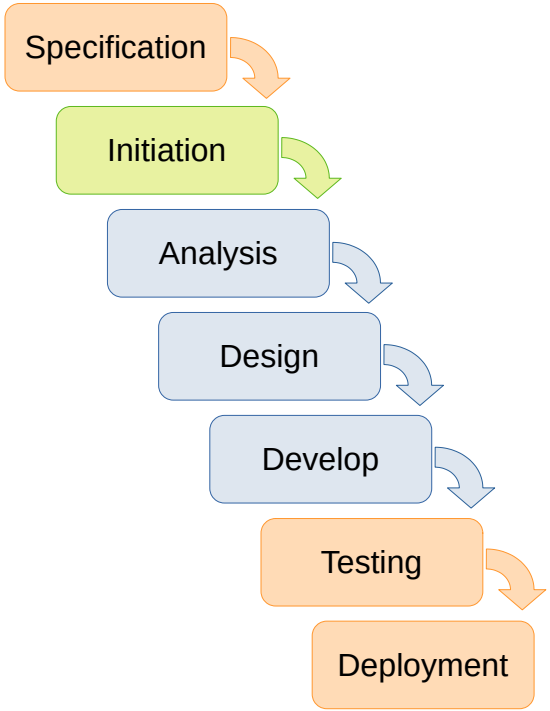
- **lack of in-house resources:** freeing in-house team to focus on FAIR-accelerator-specific tasks.

1) Mitigating Risks → ‘agile’ and ‘lean’ Framework Contract
   - general FAIR contract designed for established technologies but unsuitable for SW development
2) Software Quality and Efficiency mirrors Communication Structure (→ M. Conway’s Law)
   - **collaborations must be based on trust + open and transparent communication**
3) Public Documentation of Technologies & Outreach
   - fostering seamless collaboration & attract a diverse talent pool
   - break free from vendor lock-in, circumvent single service-provider problem

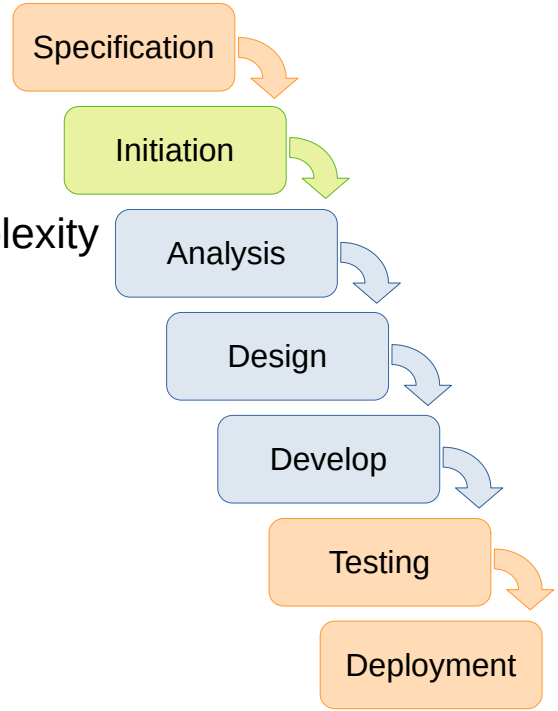
Creating Shared Value through creating public ‘clean’ and ‘lean’ Code-Base

- efficient response to 24h/7 operational needs and sustainable long-term maintenance
- FAIR as technology forge: develop technologies critical to FAIR
  & enable industry and general public in new emerging key technologies.

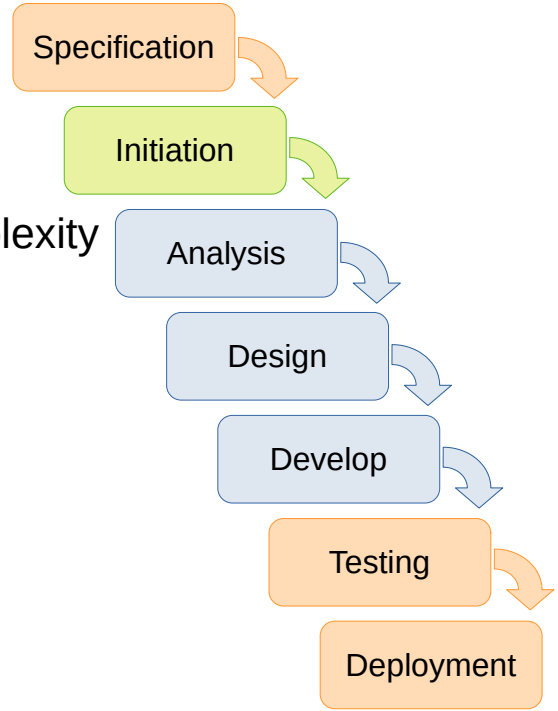
Common Problems



- “old contract”: ‘waterfall’ design process with few pros and mostly cons
  - strong tendency to over- or underspecification for software
    - overspecification time consuming for us and industry partners
    - design effort nearly identical to implementation effort
  - asymmetry between in-house and external understanding of project complexity
    - initial over- or underestimation of the project effort
  - slow feedback loops and expensive ECRs
  - “code-dump”: quality control and future maintainability

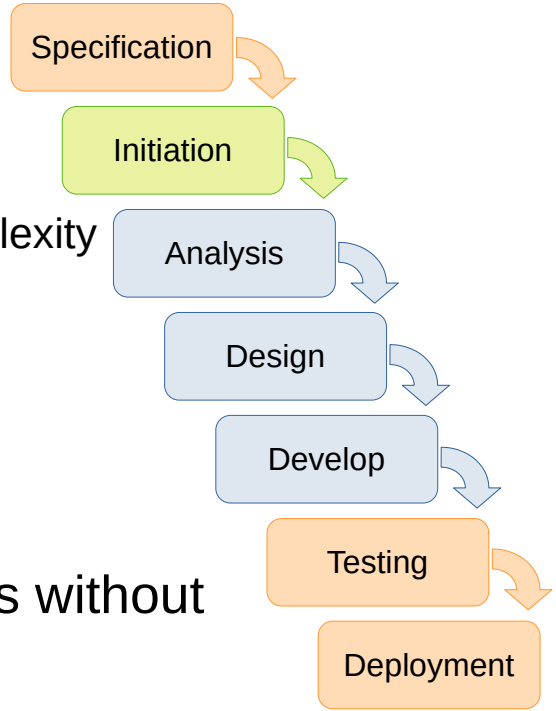


- “old contract”: ‘waterfall’ design process with few pros and mostly cons
  - strong tendency to over- or underspecification for software
    - overspecification time consuming for us and industry partners
    - design effort nearly identical to implementation effort
  - asymmetry between in-house and external understanding of project complexity
    - initial over- or underestimation of the project effort
  - slow feedback loops and expensive ECRs
  - “code-dump”: quality control and future maintainability
- closed in-house solutions: complicate onboarding



Common Problems

- “old contract”: ‘waterfall’ design process with few pros and mostly cons
  - strong tendency to over- or underspecification for software
    - overspecification time consuming for us and industry partners
    - design effort nearly identical to implementation effort
  - asymmetry between in-house and external understanding of project complexity
    - initial over- or underestimation of the project effort
  - slow feedback loops and expensive ECRs
  - “code-dump”: quality control and future maintainability
- closed in-house solutions: complicate onboarding
- hard to assess qualities and prior work of new industry partners without reliable public track record



New ‘agile’ and ‘lean’ Framework Contract (dt. “Rahmenvertrag”)

shout-out @
F. Arndt, G. Harks,
J. Schmidt

New ‘agile’ and ‘lean’ Framework Contract (dt. “Rahmenvertrag”)

- enables fast and lightweight dispatch of projects to a pre-qualified pool of industry partners.
- pool selection process:
  - demonstrated skills and expertise proven by public open-source track record
    - quality of work & sustainability (actively maintained projects vs code-dump)
    - style of community engagement

shout-out @
F. Arndt, G. Harks,
J. Schmidt

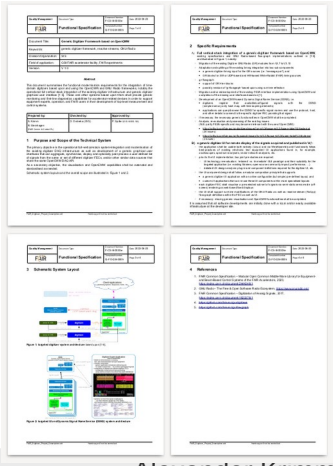
New ‘agile’ and ‘lean’ Framework Contract (dt. “Rahmenvertrag”)

- enables fast and lightweight dispatch of projects to a pre-qualified pool of industry partners.
- pool selection process:
  - demonstrated skills and expertise proven by public open-source track record
    - quality of work & sustainability (actively maintained projects vs code-dump)
    - style of community engagement
- short-form legal contract + agile process description (10 + 5 pages)
  - small, re-usable, and describes the collaborative process
  - live iterations: ‘change for free’ policy
  - incentive for efficiency: ‘early finish’ & ‘risk share’

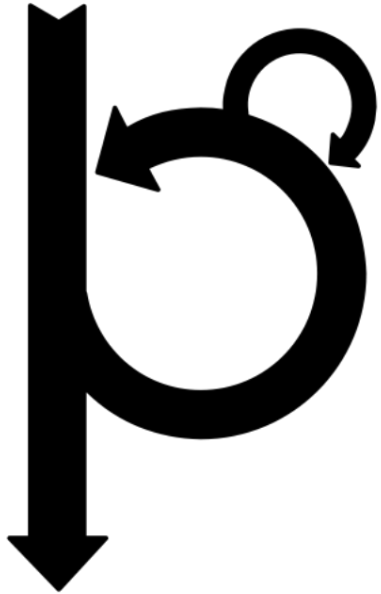
shout-out @
F. Arndt, G. Harks,
J. Schmidt

New ‘agile’ and ‘lean’ Framework Contract (dt. “Rahmenvertrag”)

- enables fast and lightweight dispatch of projects to a pre-qualified pool of industry partners.
- pool selection process:
  - demonstrated skills and expertise proven by public open-source track record
    - quality of work & sustainability (actively maintained projects vs code-dump)
    - style of community engagement
- short-form legal contract + agile process description (10 + 5 pages)
  - small, re-usable, and describes the collaborative process
  - live iterations: ‘change for free’ policy
  - incentive for efficiency: ‘early finish’ & ‘risk share’
- project description (2-10 pages, by FAIR)
  - outlines scope of work that should be performed
  - details provided in referenced public resources

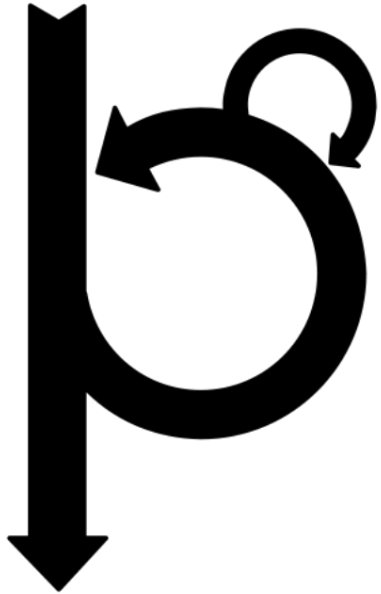


shout-out @
F. Arndt, G. Harks,
J. Schmidt

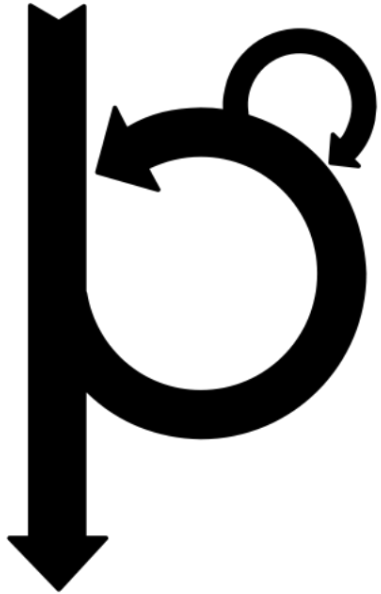


Alexander Krimm, A.Krimm@gsi.de, 2023-10-20

- **function over form**
  - focus on design, function and intended use
  - ... rather than overspecified implementation details that are impossible to know at project start

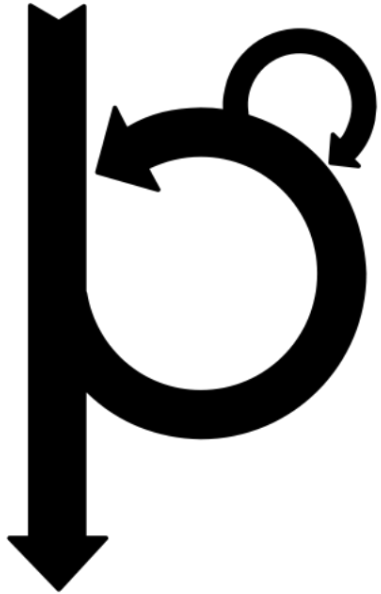


- **function over form**
  - focus on design, function and intended use
  - ... rather than overspecified implementation details that are impossible to know at project start
- **prefer building upon or extending existing projects** rather than creating new ones
  - sharing of expertise and ideas
  - spread mainteinance on more shoulders



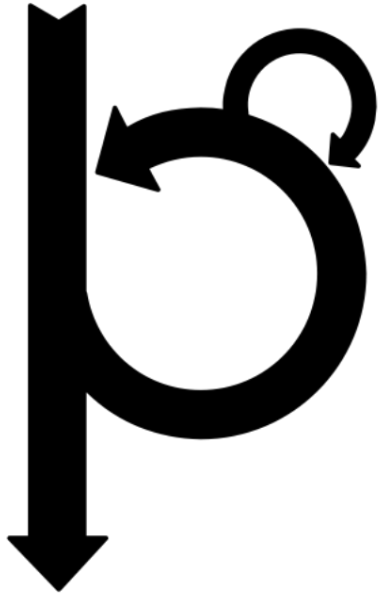
Co-Development and Continuous Iteration & Communication

- **function over form**
  - focus on design, function and intended use
  - ... rather than overspecified implementation details that are impossible to know at project start
- **prefer building upon or extending existing projects** rather than creating new ones
  - sharing of expertise and ideas
  - spread mainteinance on more shoulders
- **co-development**
  - lightweight specification + iterative process
  - allow re-priorisation and agree on a fair sharing of risks



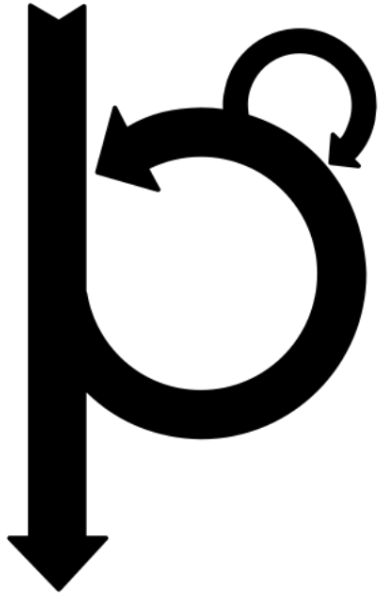
Co-Development and Continuous Iteration & Communication

- **function over form**
  - focus on design, function and intended use
  - ... rather than overspecified implementation details that are impossible to know at project start
- **prefer building upon or extending existing projects** rather than creating new ones
  - sharing of expertise and ideas
  - spread mainteinance on more shoulders
- **co-development**
  - lightweight specification + iterative process
  - allow re-priorisation and agree on a fair sharing of risks
- **shared code ownership**
  - early integration of individual functionalities
  - review process of small functional units
  - code and development process in the open as much as possible
    - for direct communication → document results publicly

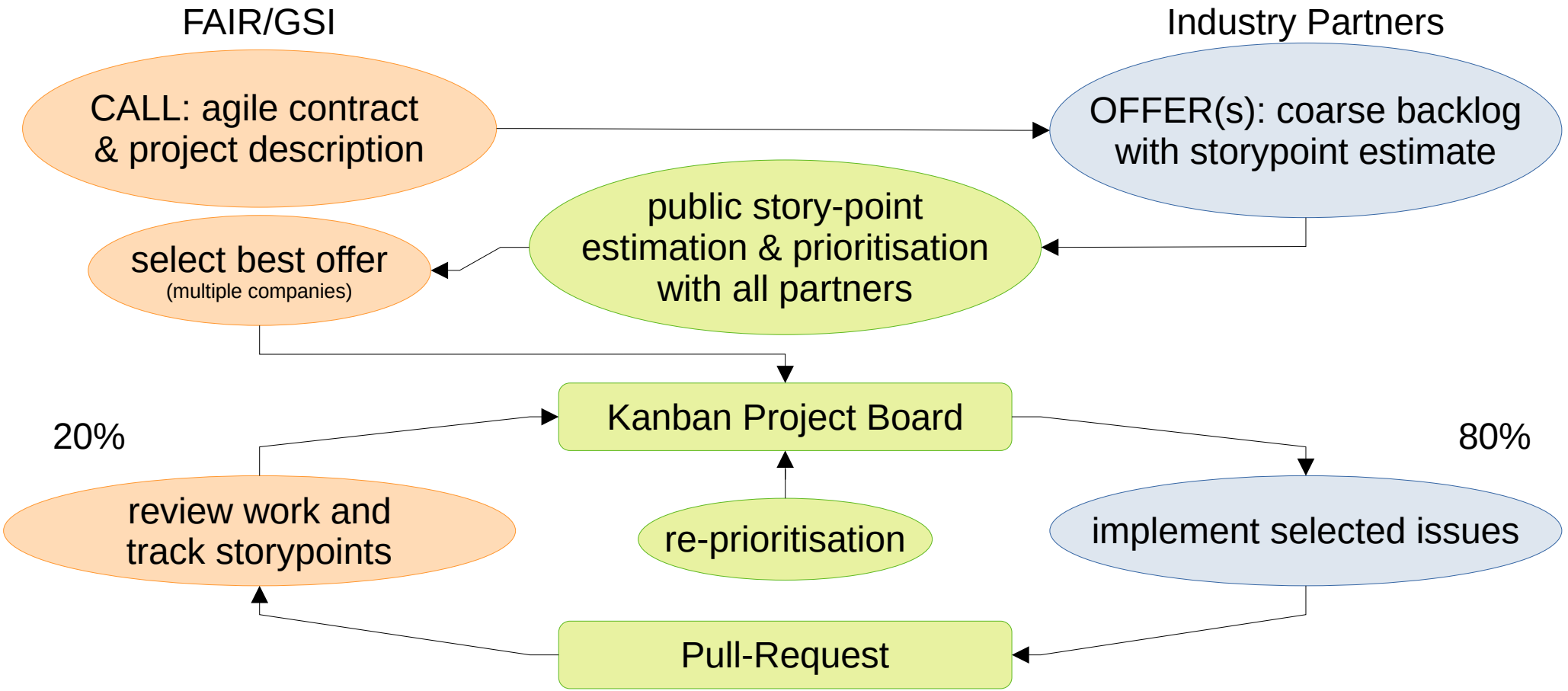


Co-Development and Continuous Iteration & Communication

- **function over form**
  - focus on design, function and intended use
  - ... rather than overspecified implementation details that are impossible to know at project start
- **prefer building upon or extending existing projects** rather than creating new ones
  - sharing of expertise and ideas
  - spread mainteinance on more shoulders
- **co-development**
  - lightweight specification + iterative process
  - allow re-priorisation and agree on a fair sharing of risks
- **shared code ownership**
  - early integration of individual functionalities
  - review process of small functional units
  - code and development process in the open as much as possible
    - for direct communication → document results publicly
- **collaboration must be based on trust and open communication**
  - communicate early on wrong estimates

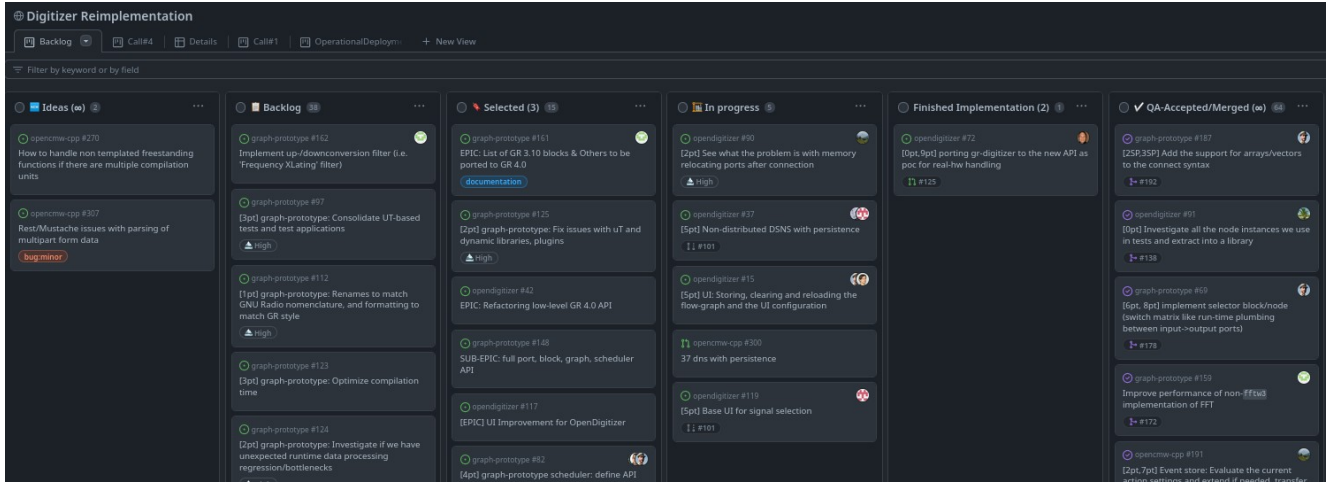


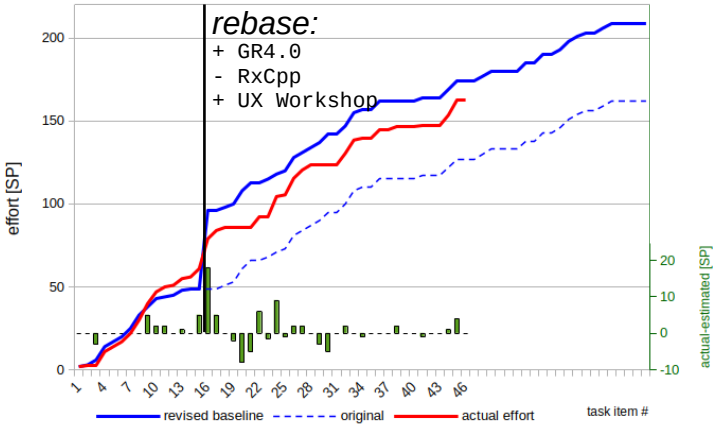
Joint Open Source Development - Overview

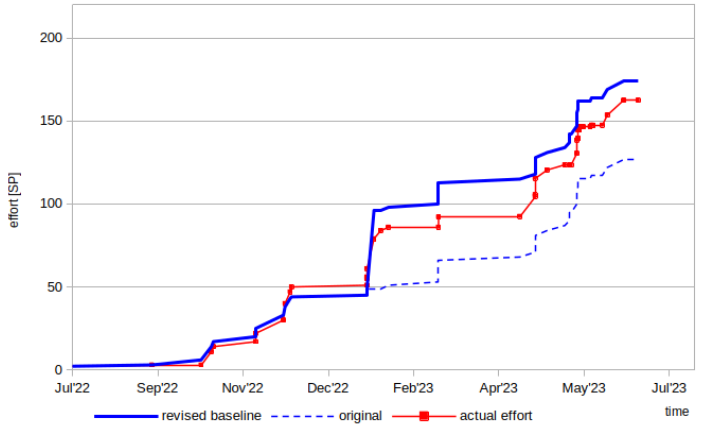


Backlog and Story-Point Estimation

- story-point: one unit of work, approximately one workday
- [estimated, actual] tracking on kanban board issues
  - functional and financial records
  - but more important: continuous feedback and improvement on shared understanding of task and project complexity



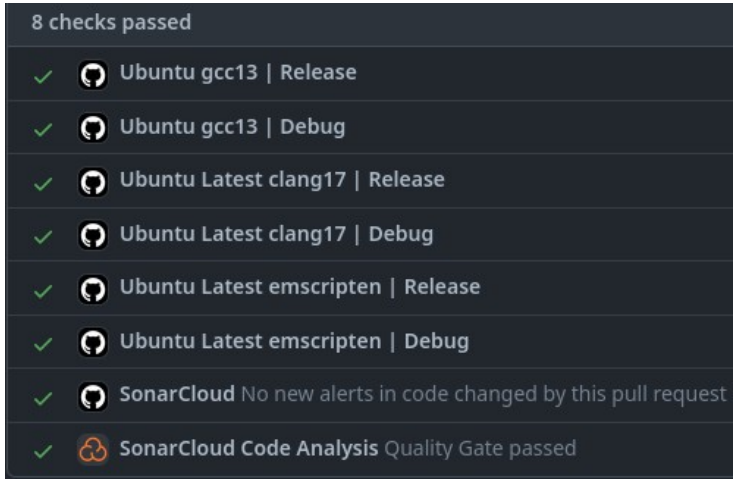




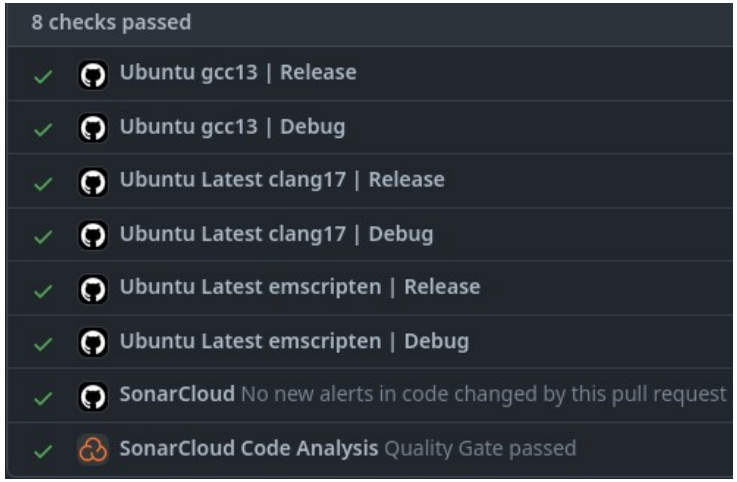
Alexander Krimm, A.Krimm@gsi.de, 2023-10-20

- issues on the Kanban board have a predefined **deliverable** and story-point **estimate**. Once done,
  - reviewed by GSI/FAIR,
  - deliverable merged, and
  - story points get tracked (triggers payment)

- issues on the Kanban board have a predefined **deliverable** and story-point **estimate**. Once done,
  - reviewed by GSI/FAIR,
  - deliverable merged, and
  - story points get tracked (triggers payment)
- PR-**reviews** are assisted by CI:
  - must build without compiler warnings,
  - must pass static code analysis tools (aka. ‘linters’),
  - must pass unit tests and code coverage criteria,
  - automated code formatting, ...



- issues on the Kanban board have a predefined **deliverable** and story-point **estimate**. Once done,
  - reviewed by GSI/FAIR,
  - deliverable merged, and
  - story points get tracked (triggers payment)
- PR-**reviews** are assisted by CI:
  - must build without compiler warnings,
  - must pass static code analysis tools (aka. ‘linters’),
  - must pass unit tests and code coverage criteria,
  - automated code formatting, ...



- for big differences between estimated and used story points, discuss causes and possible ways to get back on track

Example collaborations

Example collaborations

- digitizer framework + GNURadio + KDAB (3 x 180SP, 2019-ongoing)

Example collaborations

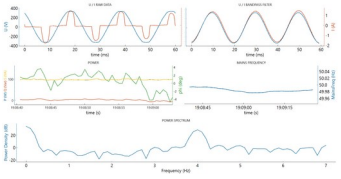
- digitizer framework + GNURadio + KDAB (3 x 180SP, 2019-ongoing)

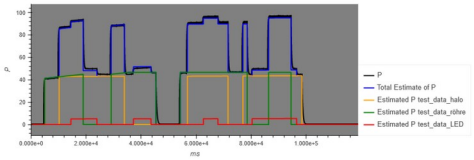
OpenCMW

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

KDAB

- AI-based pulsed-power monitoring + Infoteam AG (2 x 90SP)





infoteam
software

Example collaborations

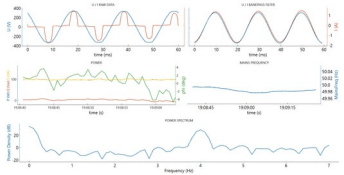
- digitizer framework + GNURadio + KDAB (3 x 180SP, 2019-ongoing)

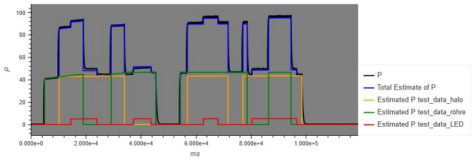
OpenCMW

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

KDAB

- AI-based pulsed-power monitoring + Infoteam AG (2 x 90SP)

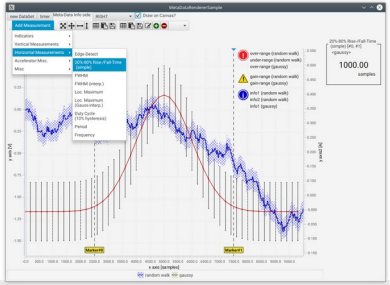




infoteam
software

- chart-fx (Java charting library) + HEBI Robotics (20SP)

ChartFx



HEBI
ROBOTICS

Summary

Why have external Partners at all?

- **lack of in-house resources:** freeing in-house team to focus on FAIR-accelerator-specific tasks.

1) Mitigating Risks → ‘agile’ and ‘lean’ Framework Contract
   - general FAIR contract designed for established technologies but unsuitable for SW development
2) Software Quality and Efficiency mirrors Communication Structure (→ M. Conway’s Law)
   - **collaborations must be based on trust + open and transparent communication**
3) Public Documentation of Technologies & Outreach
   - fostering seamless collaboration & attract a diverse talent pool
   - break free from vendor lock-in, circumvent single service-provider problem

Creating Shared Value through creating public ‘clean’ and ‘lean’ Code-Base

- efficient response to 24h/7 operational needs and sustainable long-term maintenance
- FAIR as technology forge: develop technologies critical to FAIR
  & enable industry and general public in new emerging key technologies.



Thanks for your Attention!

Do you use/develop/manage open-source projects?
What are your experiences and approaches?

Alexander Krimm, SIS100/SIS18
Open-Source Software Development with Industry, 1st Workshop on Open Science @ GSI/FAIR 20.10.2023

Finland France Germany India Poland Romania Russia Slovenia Sweden UK