A new Software Training Model at Belle II

Kilian Lieret[1], Gian Luca Pinna Angioni[2], Moritz Bauer[3], Michel Bertemes[4], Sviatoslav Bilokin[1], Angelo Di Canto[5], Giulia Casarosa[6], Racha Cheaib[7], Sam Cunliffe[7], Nathalie Eberlein[1], Michael Eliachevitch[8], Marcela Garcia[9], Phil Grace[10], Daniel Greenwald[11], Oskar Hartbrich[12], Chia-Ling Hsu[13], Ilya Komarov[7], Thomas Kuhr[1], Yaroslav Kulii[1], Frank Meier[14], Marco Milesi[15], Alejandro Mora[16], Giacomo De Pietro[17], Markus Prim[8], Martin Ritter[1], Pascal Schmolz[1], Umberto Tamponi[2], Francesco Tenchini[6], Michel Villanueva[7], Hannah Marie Wakeling[18] and Xing-Yu Zhou[19]

[1] Ludwig-Maximilians-Univ. München (LMU), Munich, Germany
[2] INFN Torino, Torino, Italy
[3] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[4] Institute of High Energy Physics, Austrian Academy of Sciences, Vienna, Austria
[5] Physics Department, Brookhaven National Laboratory, Upton, U.S.A.
[6] INFN and Univ. Pisa, Pisa, Italy
[7] Deutsches Elektronen-Synchrotron (DESY), Hamburg, Germany
[8] Univ. of Bonn, Bonn, Germany
[9] Centro de Investigacion y de Estudios Avanzados del Instituto Politecnico Nacional, Mexico
[10] Univ. of Adelaide, Adelaide, Australia
[11] Technical Univ. of Munich (Technische Universitaet Muenchen), Garching, Germany
[12] Univ. of Hawaii, Honolulu, U.S.A.
[13] Univ. of Sydney, Sydney, Australia
[14] Duke University, Durham, U.S.A.
[15] Univ. of Melbourne, Melbourne, Australia
[16] University of Tokyo, Tokyo, Japan
[17] INFN Roma Tre, Roma, Italy
[18] McGill Univ., Montreal, Canada
[19] LiaoNing Normal University (LNNU), Dalian, China

E-mail: `kilian.lieret@posteo.de, software-doc@belle2.org`

Abstract. The physics output of modern experimental HEP collaborations hinges not only on the quality of its software but also on the ability of the collaborators to make the best possible use of it. With the COVID-19 pandemic making in-person training impossible, the training paradigm at Belle II was shifted from periodic workshops towards guided self-study.

To that end, the study material was rebuilt from scratch as a series of modular and hands-on lessons tightly integrated with the software documentation using Sphinx. Each lesson contains multiple exercises that are supplemented with hints and complete solutions. Rather than duplicating information, students are systematically taught to work with the technical reference documentation to find the important sections for themselves. Unit tests ensure that all examples work with different software versions, and feedback buttons make it easy to submit comments for improvements.

1. Software training

Each year, around one hundred new members join the Belle II experiment. Among these are students working on their Bachelor's, Master's, or Ph.D. projects and more senior physicists. While their previous experience and knowledge vary greatly, everyone shares one goal: to make progress with their projects as soon as possible. At the same time, the amount of information collected in the various spaces of the collaboration is so overwhelming that the proverb "to drink from a firehose" might best describe the learning experience of many newcomers.

To ensure a smooth start for everyone, newcomers need dedicated training and support from more experienced members. This requirement particularly applies to the experiment-specific software that is central to analyzing physics data.

Training events for beginners are also one of the best opportunities to raise awareness of best practices: after newcomers get started with their scientific projects their focus tends to narrow and they become more challenging to reach. General recommendations and well-written code examples in the tutorials can significantly improve preparation for long-term success.

2. Pivoting to a self-study friendly training model

The recent remodeling of the Belle II software training material was triggered by the Covid-19 pandemic, which made in-person training events impossible. It has been shown that online events can replace most aspects of in-person events. However, on a more general note, focusing on training to occur primarily in the form of live workshops (online or in-person) rather than focusing on optimizing material for self-study is always subject to the following drawbacks:

- **Divergent needs:** The different levels of seniority, previous experience, and knowledge result in very different learning speeds, which require different sessions/tracks and more personpower to run the event.
- **Logistics, funding and the environment:** For in-person events, the required travel logistics and funding can limit both the number of attendees and the frequency of training events. The amount of air travel required also causes a significantly larger ecological footprint.
- **Scheduling:** Due to the different academic schedules of the more than 100 institutions at Belle II, newcomers join throughout the year. So far, logistical and personpower considerations have limited the number of training events to three events per year. As a result, most newcomers were forced to start their research long before attending. This causes an even more significant divergence of previous experience and diminishes the efficiency of the training.
- **Duration:** Logistics and personpower limit the training events to several days. As a result, the information presented at the events is necessarily very compressed and can quickly become overwhelming.

Taking the large-scale disruption caused by the pandemic as an opportunity, we decided to pivot the focus of the training group on material that allows efficient self-study.

3. Challenges for training material

In general, the training material is subject to the following challenges:

- **Versioning:** The analyst-facing interface of the Belle II software [1] is still evolving. New software versions can introduce backward-incompatible changes or change the recommendations and best practices.
- **Testability:** Wherever possible, code snippets in the lessons should be tested automatically. This is particularly important when changing the recommended software versions. Requiring passing tests might also be an incentive to keep the material maintainable and

on-topic, reducing the aggregation of outdated legacy material and other forms of software erosion.

- **Maintainability and Sustainability:** Lessons should be consistent, stable, and easy to update. This is an issue for more complicated code snippets incrementally built up within a lesson or building on top of code explained in a previous lesson, as changes need to be propagated carefully.
- **Interactivity:** Exercises can greatly improve the learning experience and keep readers engaged. However, it is crucial to strike the correct difficulty level for each individual. Because of the diverse audience, this can only be achieved by providing additional optional hints for each exercise, or exercises of different difficulty levels. Complete solutions should be available for all exercises.
- **Connecting resources:** Enabling newcomers to use the existing documentation resources is one of the objectives of the training itself. At the same time, referring to other resources allows the training material to be more concise and maintainable. However, links to external material can be brittle when resources move or change.

4. General principles

To face the challenges outlined in the previous section, we adopted the following principles:

- **Didactic style:** The style of the material is inspired by the work of the Software Carpentries [2, 3, 4], the work of HSF training [5, 6] and by the LHCb Starterkit [7]. The material is organized in lessons. The beginner lessons are to be studied in order and build on top of each other, while intermediate and advanced lessons are designed to be independent. Lessons consist of verbose text, code snippets, and exercises. A selection of callout boxes provides extra information, overviews, or summaries.
- **Versioning:** The training material is hosted in the git repository of the main Belle II software (*basf2*) [1, 8]. This enforces a natural correspondence of software versions and documentation/training versions. It also makes failed unit tests block the merging of pull requests: it is impossible to merge a change if it breaks one of our lessons! Reviewers of pull requests can also require the opener to make necessary updates to the training material. In short: we avoid the training material becoming an afterthought of the software development by bundling them together. As a nice side effect, this also increases the visibility of contributions to the training material.
- **Avoiding redundancy:** Wherever possible, we refer newcomers to the API documentation and similar pages, either by making it an exercise to find a piece of information or by directly linking to it. Because the API documentation and most other technical documentation are generated with the same system and hosted in the same repository, links remain functional, even for older software and training material versions.
- **Software prerequisites:** For the basics of bash, git, and python, we refer to the training material from the software carpentries but provide additional exercises with which newcomers can test their knowledge. We further extend the python training by lessons on the pandas data analysis framework. In addition, we provide a lesson on SSH.

5. Technical details

The implementation of the training material is based on the following technical solutions:

- **Generation:** The training material is available as web pages that are rendered with the Sphinx documentation generator [9]. reStructuredText is used as markup language. We have created custom callout boxes for exercise blocks, foldable hints and solutions (Fig. 2).

- **Rollout:** The training material is developed via pull requests to the main branch of the basf2 git repository. The training material from the main branch is built on a nightly basis ("development version"). The training material corresponding to releases of the software is built from the corresponding *release branches*. Major versions branch off the main branch; new commits on these branches are published as minor and patch versions. The *recommended* training version is that of the latest release. This means that updates to the training material reach the recommended version with every major release. However, hotfixes or substantial improvements can also be cherry-picked to a release branch and then included in a minor or patch release. Alternatively, newcomers can also be pointed to the development version for specific updates.
- **Preview:** Opening a pull request (and pushing additional commits) triggers a build that runs checks and gives access to a website preview. However, these builds take between 30 and 45 minutes to complete (depending on the overall resource use). Local builds are faster (1-3 minutes per build) but require an initial compilation of (part of) the main software.
- **Code inclusion:** Generally, code snippets are kept in separate source files and are included in the lessons via sphinx directives. This has several advantages:
  - The files can usually be executed as unit tests. When unit testing is not feasible, static code-checking can still ensure some level of correctness.
  - Formatting tools can be used to enforce a consistent coding style.
  - The file can be included partially in multiple places to build a more extensive example.

  Partial code inclusion is achieved with the `start-after`/`end-before` directives of sphinx that include code after/before a particular search string is found in the file. To avoid ambiguity, we use short marker strings included in the comments as search strings rather than parts of the code itself. These comments also make it evident which parts of the code are separately included while reading the file. This code inclusion scheme avoids the use of line numbers which are brittle and cumbersome to maintain.
- **Quick feedback:** Below every lesson, a set of three expandable containers provides guidance on where to get help with further questions, report problems, and how to contribute to the lesson. In particular, a tiny google form allows newcomers to report issues with the lesson anonymously with just four clicks. Entries in this google form are regularly checked and, if actionable, converted to JIRA issues that the training group uses for internal organization. However, despite the ease of use, we have received comparatively few reports through this channel.

Contributions to our training material require basic knowledge of git, pull request workflows, reStructuredText and sphinx. Acquiring these skills can be a steep learning curve, especially for newcomers – though it is very rewarding because it teaches hands-on experience in the entire software development workflow. We hope that additional step-by-step tutorials and dedicated hackathons will help grow the number of unique contributors in the long term.

6. Training events

Week-long online training events ("StarterKits") were offered with the new material in 2020 and 2021. Help was provided via chat channels and several Q & A sessions for different time zones. Because of the completeness of the lessons provided, the need for additional guidance was very low (this was also confirmed in several surveys). For this reason, we have filled the Q & A sessions with additional content (code-along exercises and more) in the latest iteration, which participants appreciated. The last day of the StarterKit workshop contains *mentoring sessions*, in which students are paired with experienced members in small groups (up to five students per mentor). Depending on the individual dynamics, these sessions can take various forms, ranging from intense debugging and detailed questions to ask-me-anything sessions.

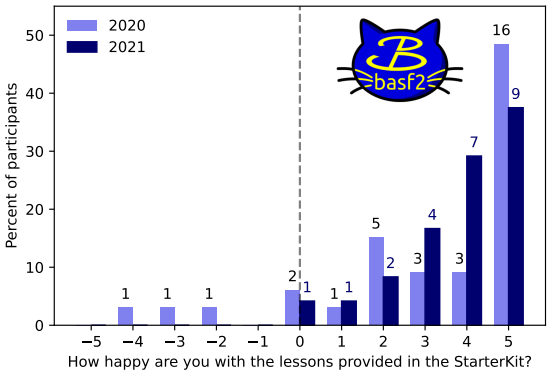

Figure 1: Customer satisfaction

Figure 2: An exercise with hints and solution (top) and an overview box (bottom)

A possible future direction is to consider more and more lessons as requirements to be completed before our events and then focus on more complicated exercises to be solved in teams with mentors.

7. Experiences

The drawback of our setup is its complexity: Slightly more complex than the setup of the LHCb StarterKit or the material of the carpentries (mainly because it is not a stand-alone repository but integrates tightly with the rest of the software) and significantly more complex than using a wiki system.

However, we think that this is more than justified by the advantages of our setup, many of which have been mentioned in the previous sections: By coupling the training material with the software and by performing unit tests, we ensure that all examples remain functional and keep the training material on the developers' agenda.

Most importantly, our material provides a very complete onboarding experience, even for newcomers who join "off-season". Lessons cover everything from basic physics knowledge, collaborative tools, and software prerequisites to submitting grid jobs. The lessons include almost 250 code snippets, 45 figures, and more than 50 overview boxes (Fig. 2). More than 200 exercises (with more than 150 hints and 190 complete solutions) make the material engaging.

We have also received very positive feedback from newcomers (Fig. 1). In the words of one of our participants: "Very solid work regarding the textbook! Congrats! Everything was very clear which significantly minimized the need for guidance (...) Software Prerequisites was the highlight of the workshop as it summarized all the necessary tools that no one really spends time on explaining thoroughly to newcomers."

The entirety of our material is publicly available at https://training.belle2.org and [8].

References

[1] Kuhr T, Pulvermacher C, Ritter M, Hauth T and Braun N (Belle-II Framework Software Group) 2019 *Comput. Softw. Big Sci.* **3** 1 (*Preprint* 1809.04299)
[2] Software Carpentry http://software-carpentry.org/ accessed on 26.02.2022
[3] Becker E and Michonneau F *The Carpentries Curriculum Development Handbook* accessed on 26.02.2022
[4] Wilson G 2016 *F1000Research* **3** 62 ISSN 2046-1402 accessed on 26.02.2022
[5] HEP Software Foundation: Training Working Group https://hepsoftwarefoundation.org/workinggroups/training.html accessed on 26.02.2022

[6] Malik S, Meehan S, Lieret K, Evans M O, Villanueva M H, Katz D S, Stewart G A, Elmer P *et al.* 2021 *Computing and Software for Big Science* **5** 22 ISSN 2510-2036, 2510-2044 (*Preprint* 2103.00659)
[7] LHCb Starterkit https://lhcb.github.io/starterkit/ accessed on 26.02.2022
[8] Public repository of the Belle II Analysis Software Framework https://github.com/belle2/basf2 accessed on 26.02.2022
[9] Brandl G Sphinx: Python documentation generator https://www.sphinx-doc.org/ accessed on 26.02.2022