Draft Version 0.91 – June 1, 2018

An HPC Certification Program Proposal Meeting HPC Users' Varied Backgrounds

Kai Himstedt[1], Nathanael Hübbe[1], Julian Kunkel[2], and Hinnerk Stüben[1]

[1] Universität Hamburg
[2] Deutsches Klimarechenzentrum
[3] Technische Universität Hamburg

Acknowledgements

The authors acknowledge the discussion with Hendryk Bockelmann[2], Michael Kuhn[1], Thomas Ludwig[1,2], Stephan Olbrich[1], Matthias Riebisch[1], Sandra Schröder[1], and Markus Stammberger[3]

This work was supported by the German Research Foundation (DFG) under grants LU 1353/12-1, OL 241/2-1, and RI 1068/7-1.

1 Introduction

Computing power and complexity of HPC systems are steadily increasing. This leads to an increasing demand for a good education of their users so that they can use such systems adequately. A special challenge is to provide users with skills according to their scientific backgrounds and specific demands in terms of the usage of the system. Users in the role of testers, who want to simply run a parallel program for benchmark purposes, must e.g. have a solid knowledge of operating system basics and should be able to use a workload manager like SLURM [SLUR 17] or TORQUE [TORQ 17], but in general they do not need a deeper understanding of the technical refinements of the parallelization of the program. A user who wants to develop a parallel program will usually already be able to use the operating system and the workload manager but will need further skills to apply parallelization techniques like OpenMP [OpMP 17] or GPU-computing based on CUDA [NVID 17] at the intra-node level, MPI [MPI 17] at the inter-node level or even combinations of such techniques in the sense of a hybrid or multi-level approach.

In recent years, the growing demands to improve the HPC education are in the research focus of many projects. Rüde, for example, in the final report on exascale education in the context of the "European Exascale Software Initiative 2" [Cordi 18], describes strengthening of HPC education as an important subfield of computational science and engineering (CSE) [Rüde 15]. The urgent demand for an appropriate HPC education is also indicated by the efforts of the HPC advisory council [HPCA 17] to push it, e.g. by offering workshops and releasing best practice as well as case studies. Therefore, it comes as no surprise that recently initial results of a Scientific Computing World HPC readership survey have shown that "... training and support for HPC resources are the number one concern for both those that operate and manage HPC facilities and researchers using HPC resources." [SCW 17].

Establishing an HPC Certification Program is a central part[1] of the joint Performance Conscious HPC (PeCoH) project. In April 2017, the three Hamburg compute centers involved in PeCoH, German Climate Computing Center (DKRZ), Regional Computing Center at the Universität Hamburg (RRZ), and Computer Center at the Technische Universität Hamburg (TUHH RZ) started the Hamburg HPC Competence Center (HHCC) as a virtual institution and central contact point for their users [HHCC 17a]. HHCC will also serve as an open-for-all education platform for HPC knowledge and competences. Our HPC Certification Program approach takes the users' varied backgrounds (e.g. research area and prior knowledge) into account and focuses on performance engineering to enable them to achieve further speedups for parallel applications with efficient utilization of the HPC resources. The performance engineering aspect is of particular importance because, according to our experience at DKRZ, RRZ, and TUHH RZ, support requests are currently dominated by problems at the level of getting things to work, i.e. getting a parallel job to run. Users in this situation are far from being aware of using the expensive HPC resources appropriately.

The paper is organized as follows: In Section 2 the classical approaches for HPC education are sketched. In Section 3 our innovative approach of an HPC Certification Program will be presented, which is based on defining HPC skills a user must have to perform certain tasks like testing, building or developing parallel programs in an HPC environment. The project's progression as well as the data structures and technical modeling to define the hierarchical dependencies of the skills are also handled in this section. Finally, the major insights are concluded in Section 4, and some future work is pointed out in Section 5. The Appendix contains a detailed list of all skills we have identified for the HPC Certification Program so far.

2 Classical HPC Education

A good user education has traditionally been important, because it leads to cost reductions in the operation at compute centers by reducing efforts for user support and by more efficiently used HPC resources by well-trained users. From our observations at DKRZ, RRZ, and TUHH RZ, new users without a proper HPC education often use only the defaults of the respective workload manager for selecting HPC resources such as main memory or CPU time and often do not explicitly select an appropriate batch queue to submit their jobs, while a user with an adequate HPC education will take meaningful estimates into consideration to avoid reserving unnecessary amounts of HPC resources, long waiting times for the job start, or reaching the runtime limit of his job. At the same time the productivity of the users will be increased, because they feel comfortable in using an HPC system. This leads to a typical win-win situation.

Institutions which operate HPC systems usually offer regularly recurring teaching events about general aspects of Supercomputer hard- and software architectures and parallel programming at beginners' level as well as higher levels. For some time now, there have also been joint efforts to support the HPC education in Europe. The education and training strategy at the Barcelona Supercomputing Centre (BSC) as outlined in [Sanc 15] may serve as an example: as part of the Partnership for Advanced Computing in Europe (PRACE), BSC is working on the development of an appropriate European HPC professional training curricula. Classical HPC education is based on lectures, tutorials, and workshops addressing

[1] Further goals of PeCoH are e.g. to develop models to estimate the costs of batch jobs in order to give HPC users feedback indicating the impacts of running non-optimized workloads, and to develop analysis tools to (automatically) identify performance issues caused by well-known configuration mistakes in job scripts.

the various HPC topics. An HPC lecture usually involves a teacher presenting topics and concepts related to a course addressing HPC topics to users enrolled in that course and has a rather static character. An HPC tutorial is typically run in smaller groups and allows discussion of the content and interaction with other users. However, the general procedure stays rather static, which also applies to HPC workshops where users typically acquire HPC skills by involving more hands-on learning activities.

Nowadays, it is very simple to publish a live or recorded lecture on an online platform, which gives users the possibility to watch the video where and when they like. Tutorials commonly make the content (additionally) available online via the internet (see [LLNL 17] for an example). The interactive aspect of a classical tutorial may suffer, but that can be more than compensated by the improved accessibility of the hyperlinked content. In addition there are hybrid approaches. Zarestky and Bangerth [ZaBa 14], for example, performed an experiment to teach HPC with a so-called flipped classroom format that requires students to watch content videos before coming to class, thus freeing time in class. Based on qualitative data Zarestky and Bangerth report positive results in terms of being able to use the time in class efficiently, and instructors and students enjoyed the new format. Reflecting the workshop idea, there exists online content with a focus on practical HPC examples showing how to get things to work (e.g. [CAC 17]). The Extreme Science and Engineering Discovery Environment (XSEDE), a virtual organization to support open research, helps their users among other things by an online system to train the usage of an HPC system, structuring the corresponding information on their website by the help of major topics like "Getting Started", "Working with the System", "Visualization Resources", and "HPC System Resources". The user can select additional information about each topic to navigate within the content [XSED 17]. There are also Websites offering (online) HPC learning material (e.g. [FuLe 18], [PRACE 18]). However, sophisticated Web-based E-learning systems which cover the users' varied backgrounds and their individual learning progresses do not exist – to the best of our knowledge – for teaching HPC competences.

In addition to the benefits of using a modern Web-based approach to present the HPC content in a more dynamic and, if needed, multimedia based way, there are ideas to use computing resources more generally for additional HPC education purposes. Holmes and Kureshi [HoKu 15], for example, reported – against the background of a shortage of HPC skills and available HPC training in the UK – experiences using recycled laboratory PCs to build cluster systems for educational purposes. Not only can the students use the clusters for experiments, but the challenge to build these laboratory clusters had a positive impact in that it encourages them to search for information from a variety of sources in order to complete the building tasks, and that developed their skills and confidence in the process. Czarnel [Czar 14] presents a successful middleware approach including a Web-based interface to support easy access to HPC systems for HPC novices by hiding the queuing systems. Suh et al. [Suh$^+$ 16] adopt an approach which rather focuses on encapsulating simulation systems behind a user-friendly graphical user interface (GUI) supporting scientific workflows. This system is also made to support the education of students, but rather in the field of computational science and engineering (CSE) than in the field of HPC competences.

Summing up, online platforms for HPC education are successfully used in practice and provide great potential. However, in contrast to other areas of information technology (IT), where certificates are often used to prove IT skills[2] of the users, in the field of HPC neither

[2] There is a certification program for various levels of Linux system administration skills from the Linux Professional Institute [LPI 17] and a certification program for general (personal) computer skills from the European Computer Driving Licence [ECDL 17a] organization, which could serve as representative examples here.

commonly accepted standards exist, nor a certification program for the education. If a scientific institution provides learning material it will be determined by the special demands of the respective institution and its specific HPC environment. Therefore, this content will only cover a very small part even of basic HPC skills and a user with a lack of basic skills will presumably have difficulties to readily use other HPC systems. These are the issues addressed by our proposal for an HPC Certification Program.

3 New Approach

Living in the age of so-called digital natives, one might suppose that computer skills are picked up intuitively. The ECDL organization notes, however, that this is not the case for basic computer skills and the idea of digital natives is a dangerous fallacy that risks leaving young people without the competences they need for the workplace, and risks leaving businesses without the skilled employees they need [ECDL 17b]. It can be assumed that this is all the more true for the complex field of HPC.

The ambitious EuroLab-4-HPC project, funded in the context of the Horizon 2020 research and innovation programme, focuses on developing a structured HPC systems curriculum and training practices based on (online) courses [EURO 18a]. As a project result it is shown how the courses can also be mapped to other degree programs (e.g. physics, mathematics) at the master's level or how they can be used for a single year's program that is Bologna-aligned ([EURO 18b] p. 12). Certificates are clearly of less significance compared for example to a master's or Phd degree, but on the other hand university degrees, with their rather great scope and possibly more national character, do not attest knowledge or skills of specific and topical technologies. This training gap can be filled ideally by the help of certification programs.

We named our HPC Certification Program "HPC-Führerschein" (HPC driving licence in English) to point out that users should have a set of validated skills before they use an HPC environment for their research. Another analogy is the transferability of skills: Anyone who is able to drive a certain type of passenger car is able to drive any other passenger car, and an HPC user who has gained the skill to use a workload manager like TORQUE will be able to use SLURM after short period of additional training, and vice versa.

Before the new approach of the HPC Certification Program is presented in more detail, we will introduce a set of terms:

Skill: The abilities and the knowledge specified in the skill description

Certified Skill: Skill of a user validated by an exam

Content: Learning material enabling the user to gain certified skills

Content provider: Institution that provides content

Exam: Process to validate a user's skill based on multiple-choice tests

Certificate definition: Set of skills as specified in the description of the certificate

Certification provider: Institution that suggests certificate definitions and corresponding exams

Certification board: Institution that establishes accepted certificate definitions and corresponding exams

Certificate: Document based on certified skills according to the corresponding certificate definition

In our approach the certificate definition is separated from content providing. While the certification board has the role of a (virtual) central authority, the learning material can be provided by different content providers, e.g. by different scientific institutions. This is comparable to the concept of a central high school graduation exam (Zentralabitur in German), where the examination is created by a central organization while the pupils are prepared for the exam by their schools. Since the start of the PeCoH project, when we had the role as certification provider as well as the role as content provider for basic HPC skills[3], we welcomed the collaboration with other scientific institutions to establish generally recognized certificate definitions. Essentially, it is at the discretion of a content provider to decide which learning material is most appropriate to teach a skill. That offers freedom and flexibility in creating the learning content. We assume that collaborating scientific institutions will complement each other in producing content.

3.1 Previous Work

We started our development of the HPC Certification Program with the classification of HPC topics which were relevant to the three compute centers (DKRZ, RRZ, and TUHH RZ) involved. We initially identified four top level competences: "HPC Knowledge", "Use of the HPC Environment", "Performance Engineering", and "Software Engineering for HPC" as shown in Figure 1.

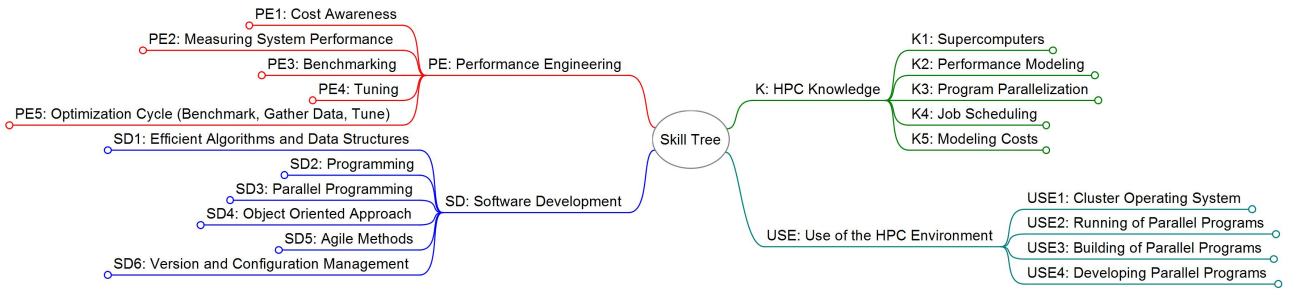


Fig. 1: Top Level Competences

We presented a poster of the current state and goals in the PeCoH project at the ISC 2017 [Kunk+ 17] and distributed a handout containing the initial classification of HPC competences and the work in progress of our HPC Certification Program [HHCC 17b], which was one of the major topics of the poster. We also presented the idea of the HPC Certification Program at the Flexible Framework for Energy and Performance Analysis in HPC Centers (FEPA) workshop [FEPA 17]. At both events we received positive feedback in several meetings and discussions, which underlines again the urgent demand for an appropriate HPC education at other compute and data centers. Additionally, we are hosting a mailing list for the HPC Certification Program [HHCC 17c].

[3] Within the PeCoH project we will establish all significant certification definitions. To produce content for all HPC skills listed in the Appendix we depend on the collaboration of others.

3.2 HPC Skill Tree

It is in the nature of the subject that HPC skills are generally built upon one another, which results in a tree structure for representing skills depending on sub-skills. The tree of HPC skills is a key component of our approach and has a role of a database for the HPC Certification Program. First of all, skills have unique names and contain a description of the HPC competences and knowledge that are associated to them. Furthermore, each skill is assigned to one of the four top level competences as described in the previous section and has additional attributes to describe its properties in more detail, like its special significance to a scientific domain (e.g. social sciences, natural sciences, earth sciences), the suitability for a user's role (e.g. tester, developer), or its educational level (e.g. basic, intermediate, or expert). This information allows to easily create different views of the skill tree in order to consider the users' varied backgrounds, e.g. for navigating within the skill tree by the help of a Web-based GUI using the attributes to filter the relevant information for them.

The implementation of the skill tree is based on the Extensible Markup Language (XML) [W3 17a] and a corresponding XML Schema Definition (XSD) [W3 17b]. XML is an open defacto standard to process and exchange information in heterogeneous environments. XML data is human- as well as machine-readable, which supports the shared working on the skill tree implementation: XML files can e.g. be opened and inspected by project participants with their favoured (simple) text editor. With the machine-readable property of XML it is possible to check the syntax of an XML file having been changed, with respect to the so-called well-formedness, and validate it with the corresponding XML schema definition. A further potential is the ability to process the data with sophisticated tools, e.g. parsers, in a variety of ways. Another reason we decided to implement the skill tree on the basis of XML is the variety of powerful tools and integrated development environments (IDEs) available to support such development (e.g. MissionKit [Alto 17], Stylus Studio X16 [StSt 17], or Eclipse XML Editors and Tools [Ecli 17]). Since the skill tree is of manageable size, there is no need to use a more complex database design for its representation. JSON [JSON 17] is another popular human- and machine-readable data-interchange format, which is rated a little bit more lightweight than XML, and was also worth considering to be used to implement the skill tree. While JSON focuses on the temporary exchange of data, the XML world provides a rich family of languages, which seems to offer more potential for the modelling process. If necessary, however, XML data can easily be converted to other formats like JSON (and vice versa), in particular by the help of XSLT [W3 17c].[4]

The essential data structure of the skill tree is presented in Figure 2 based on the relevant part of its XML Schema Definition.

As is typical for popular naming conventions of XML data structures, the *Skills* definition in the Figure shows that the XML data of the skill tree contains, first of all, a list of *Skill* items, i.e. the XML data contains all the nodes of the skill tree in a flat data structure. In order to describe the tree, each *Skill* that depends on other sub-skills has – besides its unique name, description, and further attributes – a list of references to these sub-skills. For example, in our design, the skill to build a parallel program, e.g. via an open source package, will at least require the skill to run a parallel program in an HPC environment and that in turn will require skills to use the command line interface of the operating system and a workload manager like SLURM or TORQUE. Unique skill names are used for this referencing to other skills in the *Skills* list.

[4] At the implementation level, we plan to use JavaScript [JaSc 17], which has a native support for JSON, to make the skill tree browsable in a Web-based GUI.

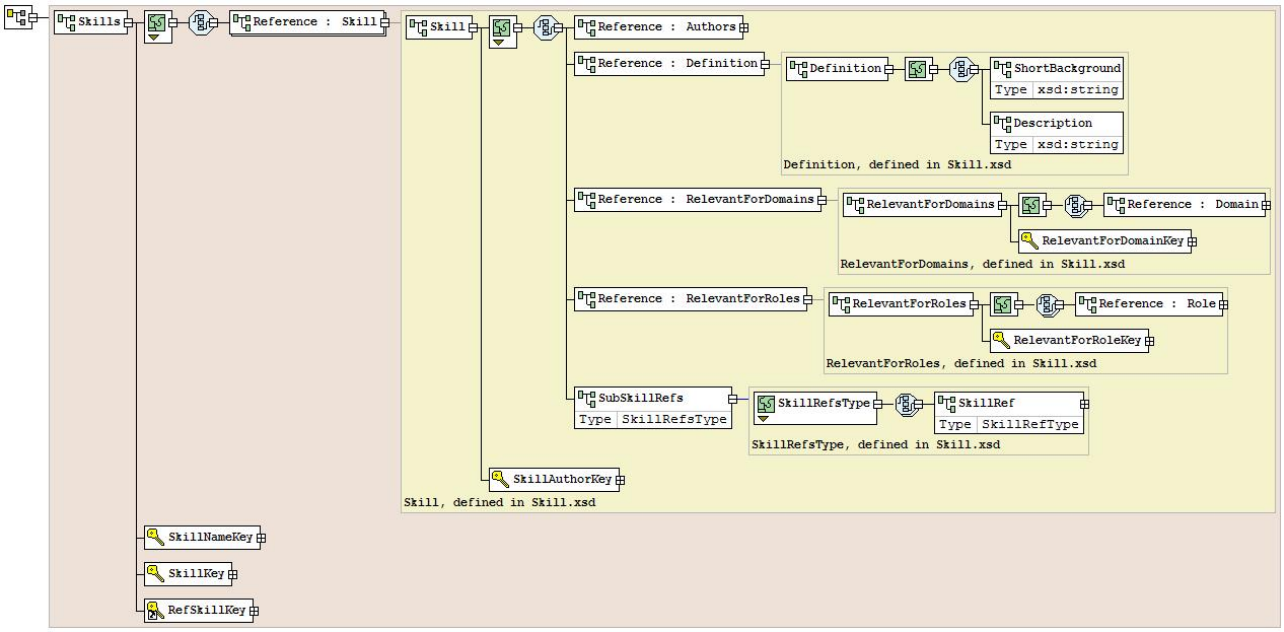


Fig. 2: XML Schema Definition for Showing the Essential Skill Tree Structure

Obviously, this data structure allows the definition of different skills depending on the same sub-skill, so, strictly speaking, the skill tree becomes a directed acyclic graph (DAG).[5] This is similar to using a `Makefile` for the well-known `make` build automation tool [Feld 79] to define the dependencies of compilation units: in C, for example, header files often contain declarations that are used (i.e. included) in different source files and other header files. The `Makefile` allows to rebuild libraries and the target program in the correct order after source code changes by the help of a depth-first search to resolve all transitive dependencies between the compilation units. Similarly, a user could aqcuire relevant skills at the leaf level of the skill tree first and than proceed to acquire skills nearer to the root. To be able to show all skills in a tree format, e.g. in a Web-based GUI, multiple references to the same skill could be resolved by presenting the more than once referenced skill several times, so, for the sake of simplicity, the DAG property shall be neglected here.

In contrast to a `Makefile` defining a single type of relationship between dependencies[6], two types of relationships are supported by the skill tree structure to define dependencies for skills: In the standard case, all skills in the list of referenced sub-skills are combined implicitly by a logical *and* operation, i.e. a skill can only be gained if *all* of its sub-skills have been gained. The second relationship is based on logical *or* operations, and allows users to gain a skill when at least *one* of the referenced sub-skills has been gained. For example, the skill to use a workload manager can be awarded to users who are able to use one of the workload managers SLURM *or* TORQUE. In practice, it will follow from context when which list type should be used in the skill definition to reference sub-skills. Within the same skill, however, it is not possible to use both types of reference lists at the same time. It would be possible to support this directly by using the composite pattern [Gamm$^+$ 95]: The basic list of implicitly *and*-combined referenced sub-skills could additionally contain lists of *or*-combined referenced sub-skills. This could be expressed in the XML Schema Definition without greater effort. But since it is easily possible to create an additional skill containing

[5] In the Appendix containing the detailed list of all skills, for the sake of simplicity as plain text format, such cross references to other skills begin with "see also ...".

[6] The time stamp of a file can be out of date in relation to the time stamps of the files it depends on. This way `make` can for example rebuild an object file if it is out of date in relation to more recently changed source files it depends on.

the list of *or*-combined referenced sub-skills and referencing to this additionally created skill in the list of *and*-combined referenced sub-skills, we preferred to keep the data structures as simple as possible.

The attributes of the skills allow to present the skill tree in a highly dynamic manner. This way users can first of all get an overview of those custom-tailored skills which they need for the HPC environment they would like to use or the parallel program they would like to speed up. However, the skill tree itself is content-free and solely describes which HPC competences have to be taught and learned. This reflects the separation of the certificate definition, which is based in our approach on skills, from the learning material that allows the user to gain the associated skills. In the sense of an E-learning environment it is possible to present a specific content in a Web-based system, which in turn maps it to the skill tree. In further stages of the project, the skill tree can be extended to support links to learning material. In this way, a single Web-based system can be used for browsing the skill tree as well as the content.

A special challenge is to determine a reasonable granularity of skills as defined by their descriptions. One can easily imagine that an increasingly finer granularity results if one attempts to dissolve the leaves of the skill tree more and more, with a skill at the leaf level finally predefining its content. At the beginning, we actually dissolved basic skills how to use the Linux command line interface to verify the practicability of our XML implementation of the skill tree. This was indeed possible without any problems, but from the fine granularity almost a one-to-one relationship results between the skill description and the related content, so that simply put each skill would have been imparted to the extent of a single presentation slide. A representative skill definition from this ad hoc example can illustrate this: the skill "Navigate the file system" was dependent on the sub-skills "Understand the file system tree", "Print name of current working directory", "Change directory", and "List directory contents". For the entire ad hoc example the definition of 59 skills was required just for describing some frequently used Linux commands (`cd`, `ls`, `less`, `cat`, `cp`, `mv`, `mkdir`, `rm`, `help`, `info`, `chmod`, `chown`, and `chgrp`).

It is obvious that a very fine granularity not only restricts the freedom in providing the content, but also makes it more difficult to define certificates because the number of skills will strongly increase accordingly. A granularity that is too coarse, such as a limitation to the top level competences shown in Figure 1, is also not useful as it would give the content providers essentially no assistance in structuring the learning material. The Appendix contains the list of all 46 skills we have identified for the HPC Certification Program so far, 35 of which are at the leaf level. We think with this granularity we found a good compromise between both extremes in order to separate skills and content. We will use up to three levels of education (basic, intermediate, and expert) to further subdivide each skill and to define the HPC competence level a user has acquired with regard to a skill. The educational levels are also shown in the Appendix. The process of subdividing requires experience and expert knowledge. The information about skill attributes and educational levels is contained in the XML description of the skill tree, which is available on our HHCC website [HHCC 18a].

3.3 Certification Modeling

The basic idea of defining certificates is to bundle a set of skills corresponding to the certificate description in order to certify – by successful exams – a user's HPC qualification. Like skills, certificate definitions have unique names and contain a description of the HPC competences and knowledge that are associated to them. While a skill is a more self-contained

unit, a certificate definition describes on a conceptual level a further view on the skills. The skill tree represents a middle-layer between the certificate definitions and the actual content.

Initially, we intended to implement the certificate definitions by a separate data structure, which was based, like the definition of the skill tree, on XML and a corresponding XML Schema Definition. But since both XML structures were so similar that their distinction served rather a conceptual purpose than a technical one, it was natural to extend the skill tree – functioning as a central database – to be able to incorporate the properties additionally required for the definition of certificates as well, instead of using a separate structure for defining certificates. At first a skill can be easily tagged for its additional suitability as an autonomous certificate definition. A user who has gained the skill to run parallel programs in an HPC environment may thus be granted a corresponding certificate. For the skill tree it was described that two types of relationships are supported (based on *and* and *or* operations) to define dependencies on sub-skills, so that a skill is gained if all of its sub-skills are gained (*and* operation) or if one of its sub-skills is gained (*or* operation). For the definition of certificates these two types of operations were supplemented by an *n out of m* relationship, so a skill is considered to be gained if at least n, or a corresponding percentage value, of its sub-skills have been gained. With this type of relationship, users can be certified, for example, for their experienced ability to use version control systems, if they gain sub-skills to use two systems from the set consisting of the Revision Control System (RCS) [Tich 85], Subversion (SVN) [TASF 17], and Git [Git 17].

4 Conclusions

There is an urgent demand to improve the users' HPC education to enable them to use the HPC resources appropriately. This will increase their productivity and at the same time reduce the costs in the operation at compute centers. While certificates are widely applied in the IT industry to testify that users have certain skills, e.g. to administer Linux systems, this is not the case for the field of HPC. With the proposal of an HPC Certification Program we try to establish a standard for the education of HPC users. In our approach we separated the certificate definitions from the providing of the learning material. By its role as a (virtual) central authority the certification board has the power to establish generally accepted certificate definitions and corresponding exams without the burden of being responsible for the content. The content can be provided in a variety of ways by various collaborating providers.

Sophisticated Web-based E-learning systems which cover the users' varied backgrounds (concerning for example research area and prior knowledge) do not exist for the HPC education. For our approach we implemented an HPC skill tree based on XML and a corresponding XML Schema Definition (XSD), which plays the role of a central database (see also Appendix). Beside its name and description, a skill in the tree has additional attributes to describe e.g. its special significance to a scientific domain. Such information can be easily used to create different views of the skill tree in order to consider the users' varied backgrounds and to give the user an overview of those custom-tailored skills which he has to acquire to pass the exams. Not only can well-trained certified users with a good knowledge of performance engineering concepts speed up their parallel programs to get their scientific results faster, but also can compute centers reduce their costs because the HPC resources will be used more efficiently.

One major challenge was to find a good compromise for the scope of the skill descriptions, in particular at the leaf level, so that a too fine granularity will not predefine the content

of a skill or an all too coarse granularity will be of no help at all for the content providers for structuring the learning material. The skill tree for our HPC Certification Program contains 46 skills (see also Appendix), which we consider to be a suitable granularity.

5 Status, Collaboration, and Future Work

The development of the HPC skill tree is nearly completed. At DKRZ, RRZ, and TUHH RZ we already have some content to teach HPC topics, which can be used to fill the content-free structure level formed by the HPC skill tree for our online education platform. We welcome suggestions from interested readers on the tree structure and the actual classification of HPC skills. Furthermore, we encourage readers to provide us with content for HPC skills and will express our gratitude by a corresponding entry in the acknowledgement area on our website. (See also Appendix for the list of skills.)

A user will have to participate in online examinations based on multiple-choice tests to gain an HPC certificate. For each HPC skill a pool of questions is developed, of which a subset is selected for each individual examination. Once the test is completed, the system will automatically assess the results and create a PDF with the certificate. At the beginning, we will manually approve the test results. Later on in the development, the individual learning progress could be stored as a part of the user account, allowing users to interrupt their exam preparation at any time and continue later to navigate seamlessly in the learning content.

It will be particularly interesting to measure the success of the certificate-based approach for the HPC education. One idea is to see if there will be less support requests of new users with simple demands for running parallel programs on the clusters at DKRZ, RRZ, and TUHH RZ. With additional surveys, the users' satisfaction with the certification program can be determined. It will also be possible to check if the performance awareness of certified users is raised, i.e. if they use the HPC resources more appropriately.

6 References

[Alto 17] Altova. *Altova MissionKit – Award-winning Suite of XML, SQL, & UML Tools*. https://www.altova.com/missionkit

[CAC 17] CAC. *Cornell University Center for Advanced Computing – Cornell Virtual Workshop*. https://cvw.cac.cornell.edu/default

[CORDI 18] CORDIS. *Community Research and Development Information Service: European Exascale Software Initiative 2 – Towards exascale roadmap implementation*. http://cordis.europa.eu/project/rcn/105840_en.html

[Czar 14] Czarnul, Pawel. Teaching High Performance Computing Using Beesy-Cluster and Relevant Usage Statistics. *International Conference On Computational Science (ICCS 2014), Procedia Computer Science. Vol.* 29 (2015):1458-1467.

[Ecli 17] Eclipse. *Eclipse XML Editors and Tools*. https://marketplace.eclipse.org/content/eclipse-xml-editors-and-tools-0

[ECDL 17a] ECDL. *European Computer Driving License – Home Page*. http://ecdl.org/

[ECDL 17b] ECDL. *European Computer Driving License – The Fallacy of the Digital Native*. http://ecdl.org/policy-publications/digital-native-fallacy

[EURO 18a] EUROLAB-4-HPC. *EuroLab-4-HPC – Home Page*. https://www.eurolab4hpc.eu/

[EURO 18b] EUROLAB-4-HPC. *D3.2 Best Practices in HPC Training*. https://www.eurolab4hpc.eu/static/deliverables/D3-2–best-practices-HPC-training.610d055cf370.pdf

[Feld 79] Feldman, Stuart I. Make – A Program for Maintaining Computer Programs. *Software Practice & Experience*. Vol. 9 (1979):255-265.

[FEPA 17] FEPA. *Flexible Framework for Energy and Performance Analysis in HPC Centers – Workshop 2017*. https://blogs.fau.de/prope/fepa-workshop-2017/

[FuLe 18] FutureLearn. *Online Course Supercomputing*. https://www.futurelearn.com/courses/supercomputing#section-topics

[Gamm$^+$ 95] Gamma, Erich, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. Boston, San Francisco, New York 1995.

[Git 17] Git. *git –fast-version-control*. https://git-scm.com/

[HPCA 17] HPCA. *HPC Advisory Council – Home Page*. http://www.hpcadvisorycouncil.com/

[HHCC 17a] HHCC. *Hamburg HPC Competence Center – Home Page*. https://www.hhcc.uni-hamburg.de

[HHCC 17b] HHCC. *Hamburg HPC Competence Center – Handout to the work in progress of the HPC Certification Program*. https://www.hhcc.uni-hamburg.de/en/files/isc2017-hpc-certification-program.pdf

[HHCC 17c] HHCC. *Hamburg HPC Competence Center – Mailing List of the HPC Certification Program*. certification.hhcc@lists.uni-hamburg.de

[HHCC 18a] HHCC. *Hamburg HPC Competence Center – Download Area*. https://www.hhcc.uni-hamburg.de/en/support/downloads.html

[HoKu 15] Holmes, Violeta, and Ibad Kureshi. Developing High Performance Computing Resources for Teaching Cluster and Grid Computing courses. *International Conference On Computational Science (ICCS 2015), Procedia Computer Science*. Vol. 51 (2015):1714-1723.

[JaSc 17] JavaScript. *JavaScript – Reference*. https://developer.mozilla.org/en-US/docs/Web/JavaScript

[JSON 17] JSON. *JavaScript Object Notation – Introducing JSON*. http://www.json.org/

[Kunk+ 17] Kunkel, Julian, Michael Kuhn, Thomas Ludwig, Matthias Riebisch, Stephan Olbrich, Hinnerk Stüben, Kai Himstedt, Hendryk Bockelmann, and Markus Stammberger. Performance Conscious HPC (PeCoH) – Project Poster. *ISC High Performance 2017 (20 June 2017)*. Frankfurt, Germany. Download via http://isc-hpc.com/isc17_ap/presentationdetails.htm?t=presentation&o=1196&a=select&ra=personendetails

[LLNL 17] Lawrence Livermore National Laboratory. *Livermore Computing Center – High Performance Computing: Tutorials*. https://hpc.llnl.gov/training/tutorials

[LPI 17] LPI. *Linux Professional Institute – Home Page*. http://www.lpi.org/

[MPI 17] MPI. *The Message Passing Interface (MPI) standard*.www.mcs.anl.gov/research/projects/mpi/

[NVID 17] NVIDIA. *CUDA Zone*. https://developer.nvidia.com/cuda-zone

[OpMP 17] OpenMP. *The OpenMP API Specification for Parallel Programming*. www.openmp.org

[PRACE 18] Partnership for Advanced Computing in Europe. *Training Portal – Training Courses*. http://www.training.prace-ri.eu/nc/training_courses/index.html

[Rüde 15] Rüde, Ulrich. *European Exascale Software Initiative 2: Deliverable D2.3 WP2 Final Report on Exascale Education*. www.eesi-project.eu/wp-content/uploads/2015/05/EESI2_D2.3_Final-report-on-exascale-education.pdf

[Sanc 15] Sancho, Maria-Ribera. BSC Best Practices in Professional Training and Teaching for the HPC Ecosystem. *Journal of Computational Science*. Vol. 14 (2015):74-77.

[SLUR 17] SLURM. *SLURM Workload Manager Overview*. https://slurm.schedmd.com/overview.html

[SCW 17] Scientific Computing World. *Training and Support Number One Concern for the HPC Community*. https://www.scientific-computing.com/news/training-and-support-number-one-concern-hpc-community

[Suh+ 16] Suh, Young-Kyoon, Hoon Ryu, Hangi Kim, and Kum Won Cho. EDISON: A Web-based HPC Simulation Execution Framework for Large-scale Scientific Computing Software. *16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid)*. IEEE Conference Publications. (2016):608-612.

[StSt 17] Stylus Studio. *X16 – Powerful XML Development*. https://www.stylusstudio.com/index.html

[TASF 17] The Apache Software Foundation. *Apache Subversion – Enterprise-class centralized version control for the masses*. https://subversion.apache.org/

[Tich 85] Tichy, Walter F. RCS – A System for Version Control. *Software Practice & Experience* 15 (1985):637-654.

[TORQ 17] TORQUE. *Torque Resource Manager*. www.adaptivecomputing.com/products/open-source/torque/

[W3 17a] W3. *World Wide Web Consortium – Extensible Markup Language (XML)*. https://www.w3.org/XML/

[W3 17b] W3. *World Wide Web Consortium – XML Schema*. https://www.w3.org/2001/XMLSchema

[W3 17c] W3. *World Wide Web Consortium – XSL Transformations (XSLT)*. https://www.w3.org/TR/xslt/

[XSED 17] XSEDE. *Extreme Science and Engineering Discovery Environment – Training*. https://portal.xsede.org/web/xup/training/overview

[ZaBa 14] Zarestky, Jill and Wolfgang Bangerth. Teaching High Performance Computing: Lessons from a Flipped Classroom, Project-Based Course on Finite Element Methods. *Workshop on Education for High Performance Computing (EduHPC) held in conjunction with SC14: The International Conference for High Performance Computing, Networking, Storage and Analysis*. New Orleans, Louisiana, November 16-21. IEEE Conference Publications (2014):34-41.

7 Appendix

At the end of the Appendix, the HPC skill tree is presented as a compact diagram.

In the following, all skills we have identified for the HPC Certification Program so far are listed in a hierarchical manner to reflect their underlying tree structure as described in Section 3. The hierarchy here is based on four top level competences: "HPC Knowledge", "Use of the HPC Environment", "Performance Engineering", and "Software Engineering for HPC". The *Description* section of a skill specifies the abilities and the knowledge a user will gain. When appropriate, some additional information may be presented in the *Short Background* section.

For the sake of simplicity, the attributes of the skills to indicate a special significance for users in dependence of their varied backgrounds (e.g. social scientist, natural scientist, earth scientist) or roles (e.g. tester, developer) are not included here. The same holds for the finer differentiation of a skill description regarding its educational level (e.g. basic, intermediate,

or expert). This type of additional information is contained in the full XML description of the skill tree, which is available on our HHCC website [HHCC 18a].

K-E HPC Knowledge
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of the field of High Performance Computing (according to the respective level)

K1-E Supercomputers
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of various system-, hardware-, and I/O-architectures used for supercomputers, i.e. computers that led the world in terms of processing capacity, and particularly in speed of calculations, at the time of their introduction, or share key architectural aspects with these computers (according to the respective level)
Knowledge of typical operation of data and computing centers (basic level)
Knowledge of the differentiation between Supercomputing and Big Data (basic level)

K1.1-E System Architectures
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of storage and compute deployments for cluster systems (expert level)
Knowledge of various system-, hardware-, and I/O-architectures used for supercomputers, i.e. shared memory systems, distributed systems, and cluster systems (basic level)
Knowledge of the typical architecture of cluster systems consisting of nodes with different roles (e.g. so-called head, login, compute, interactive, visualization nodes, etc.) (basic level)

K1.2-E Hardware Architectures
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of parallelization techniques at the instruction level of a processing element (e.g. pipelining, SIMD processing) (expert level)
Knowledge of hybrid approaches, e.g. combining CPUs with GPUs or FPGAs (expert level)
Knowledge of typical network topologies and architectures used for HPC systems, like fat trees based on switched fabrics using e.g. fast Ethernet (1 or 10 Gbit) or InfiniBand (expert level)
Knowledge of special or application-specific hardware (e.g. TPUs) (intermediate level)
Knowledge of elementary processing elements like CPUs, GPUs, many core architectures (basic level)
Knowledge of vector systems, and FPGAs (basic level) (background topic)
Knowledge of the NUMA architecture used for symmetric multiprocessing systems where the memory access time depends on the memory location relative to the processor (basic level)
Knowledge of network demands for HPC systems (e.g. high bandwidth and low latency) (basic level)

Knowledge of typical network architectures used for HPC systems, like fast Ethernet (1 or 10 Gbit) or InfiniBand (basic level)

K1.3-I I/O Architectures

Relevant for: Tester, Builder, and Developer

Description:

Knowledge of standard file systems like Ext3, Ext4, XFS, and Btrfs and distributed file systems like Lustre, and BeeGFS (intermediate level)

Knowledge of when to use data compression (intermediate level)

Knowledge of typical I/O systems used in HPC environments (basic level)

Knowledge of different types of media (e.g. tape, disk, and SSD) (basic level)

Knowledge of the differentiation between standard file systems and distributed file systems (basic level)

Knowledge of when to use local and global storage (basic level)

K1.4-B Operation of an HPC System

Relevant for: Tester, Builder, and Developer

Description:

Knowledge of the typical infrastructure of data and computing centers (basic level)

Knowledge of the typical infrastructure of data and computing centers, also against the background of economic and business aspects (basic level) (background topic)

Knowledge of administration aspects of an HPC system (basic level)

Knowledge of user support aspects (typically on different levels) (basic level)

K1.5-E Supercomputing and Big Data

Relevant for: Tester, Builder, and Developer

Short Background: In the recent past, Supercomputing as well as the analysis of Big Data are increasingly growing in importance for scientific research.

Description:

Knowledge of the differentiation between Supercomputing and Big Data (expert level) (background topic)

Knowledge of Supercomputing and Big Data (basic level) (background topic)

K2-E Performance Modeling

Relevant for: Tester, Builder, and Developer

Short Background: HPC systems are massively parallel and therefore sophisticated parallel programs are required to exploit their performance potential as much as possible.

Description:

Knowledge of how the performance of parallel programs may be assessed (according to the respective level)

K2.1-I Performance Frontiers

Relevant for: Tester, Builder, and Developer

Description:

Knowledge of the roofline model, used to provide performance estimates for parallel programs based on multi-core or accelerator processor architectures, by showing inherent hardware limitations (intermediate level)

Knowledge of the definitions for key terms like speedup, efficiency, and scalability (basic level)

Knowledge of the key measure floating point operations per second (FLOPS) for the performance of HPC systems and its pitfalls (basic level)

Knowledge of Moore's law and its significance for performance frontiers in modern HPC (basic level) (background topic)
Knowledge of Amdahl's law and its significance for performance frontiers in modern HPC (basic level)

K2.2-E Bounds for a Parallel Program
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of how performance bounds of the various components of the HPC system (e.g. CPU, caches, memory) can limit the overall performance of a parallel program (expert level)
Knowledge of how performance bounds of the various components of the HPC system (e.g. network, I/O) can limit the overall performance of a parallel program (intermediate level)

K3-E Program Parallelization
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of the typical parallelization techniques used at the intra- and inter-node level of cluster systems (basic level)
Knowledge of the causes of parallelization overheads, which eventually prevent efficient use of an increasing number of processing elements (basic level)
Knowledge of domain decomposition strategies (i.e. splitting a problem into pieces that allow for parallel computation) (basic level)

K3.1-B Parallelization Without Modifying the Source Code
Relevant for: Builder and Developer
Description:
Knowledge of the auto parallelization capabilities of current compilers (e.g. to automatically parallelize suitable loops), which are applicable at the intra-node level (basic level)

K3.2-E Level of Parallelization
Relevant for: Developer
Description:
Knowledge of hybrid parallelization approaches, combining for instance OpenMP and GPU-Computing (expert level)
Knowledge of multi-level approaches (e.g. combining OpenMP and MPI) (expert level)
Knowledge of parallelization techniques at the intra-node level (e.g. based on advanced OpenMP features and GPU-computing) (intermediate level)
Knowledge of parallelization techniques at the intra-node level (e.g. based on basic OpenMP features) (basic level)
Knowledge of the message passing paradigm based on environments like MPI, which is the de-facto standard at the inter-node level for parallelizing programs using more than a single node (basic level)

K3.3-I Parallelization Overheads
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of the overheads caused by redundant computations (intermediate level)
Knowledge of the problems of execution speed noise (OS jitter, cache contention, thermal throttling, etc.), and typical trade-offs (e.g. reducing the syn-

chronization overhead by increasing the communication overhead) (intermediate level)
Knowledge of the various overheads, i.e. overheads for communication, synchronization (basic level)
Knowledge of the problems of load imbalances (basic level)

K3.4-E Domain Decomposition
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of how to map domains to machines (expert level)
Knowledge of typical decomposition strategies to split a domain into subdomains to make it suited for parallel processing (basic level)
Knowledge of measures like surface to volume ratio (basic level)

K4-E Job Scheduling
Relevant for: Tester, Builder, and Developer
Description:
Knowledge of sophisticated scheduling principles (e.g. fair share) to achieve objectives like treating the users fair, and maximizing the utilization of the available HPC resources (expert level)

Knowledge of advanced scheduling principles (e.g. backfilling) to achieve objectives like minimizing the averaged elapsed program runtimes, and maximizing the utilization of the available HPC resources (intermediate level)

Knowledge of how workload managers control the unattended background execution of programs or jobs respectively by the help of job queues (basic level)

Knowledge of typical scheduling principles (e.g. first come first served, shortest job first) to achieve objectives like minimizing the averaged elapsed program runtimes, and maximizing the utilization of the available HPC resources (basic level)

K5-E Modeling Costs
Relevant for: Tester, Builder, and Developer
Short Background: The user's awareness of the costs related to the operation of an HPC system is raised. (basic level) For the resources of an HPC system, a distinction is made between costs for the computing elements of the supercomputer and costs for the storage system. (basic level)

Description:
Knowledge of how to assess the costs for the infrastructure of data and computing centers as well as their personnel costs (expert level)

Knowledge of economic and business aspects, e.g. break-even considerations, when personnel costs for tuning a parallel program and savings through speedups achieved are compared (expert level)

Knowledge of how to assess runtime costs for jobs (intermediate level)

Knowledge of the impact of a cluster nodes type (e.g. CPU type, main memory expansion, or GPU extensions) and of the storage media type (SSD, disk, or e.g. tape for long term archiving (LTA) purposes) on its costs (basic level)

USE-E Use of the HPC Environment
Relevant for: Tester, Builder, and Developer
Description:
Ability to use a cluster operating system as well as to run, build, and develop parallel programs (according to the respective level)

USE1-E Use of the Cluster Operating System

Relevant for: Tester, Builder, and Developer

Short Background: HPC systems are usually accessed via a command line interface (CLI). The user acquires skills to use a – generally Linux based – CLI to interact with the HPC system.

Description:

Ability to use and write shell scripts e.g. to automatically execute several commands in a row in order to automate more complex tasks (intermediate level)

Ability to use and write shell scripts e.g. to automatically execute several commands in a row that otherwise would have to be entered manually one by one and to automate simple tasks (basic level)

Ability to select the right environment setting to build programs with the proper compiler, linker, and libraries versions or to run programs (basic level)

USE1.1-I Use of the Command Line Interface

Relevant for: Tester, Builder, and Developer

Description:

Ability to use regular expressions to select or filter several items at once (e.g. file content) (intermediate level)

Ability to access local and remote files using enhanced features (e.g. via SSHFS) in remote sessions (intermediate level)

Ability to execute frequently used commands, e.g. to navigate the file system, copy, rename, and delete files, view the contents of files, and to get detailed help for the usage of a command with all its options (basic level)

Ability to use wildcards to select or filter e.g. files (basic level)

Ability to login remotely to cluster nodes using e.g. SSH with password or SSH key authentication (basic level)

Ability to access local and remote files in remote sessions (basic level)

Ability to check disk quotas commonly used to limit the amount of disk space available for the user (basic level)

USE1.2-E Using Shell Scripts

Relevant for: Tester, Builder, and Developer

Description:

Ability to read keyboard input to add interactivity to scripts (expert level)

Ability to use flow control, e.g. for conditional and/or repeated execution of statements in scripts (intermediate level)

Ability to use shell functions to break large, complex tasks into a series of small, simple tasks (intermediate level)

Ability to write robust job scripts, e.g. to simplify job submissions by the help of automated job chaining (intermediate level)

Ability to use and write shell scripts (according to the respective level)

Ability to use troubleshooting, e.g. to handle syntactic and logical errors in scripts (according to the respective level)

USE1.3-B Selecting the Software Environment

Relevant for: Tester, Builder, and Developer

Short Background: HPC systems have generally installed multiple versions of a number of key software tools and software environments. Package managers like Spack are sketched. (basic level)

Description:

Ability to select the appropriate versions for deployment to the session environment, e.g. via the so-called environment modules system (basic level)

USE2-E Running of Parallel Programs

Relevant for: Tester, Builder, and Developer

Description:

Ability to use the command line interface (expert level) (see also USE1.1-I Use of the Command Line Interface)

Ability to write robust job scripts, e.g. to simplify job submissions by the help of automated job chaining (expert level) (see also USE1.2-E Using Shell Scripts)

Ability to select the appropriate software environment (expert level) (see also USE1.3-B Selecting the Software Environment)

Ability to consider cost aspects (expert level) (see also PE1-E Cost Awareness)

Ability to measure system performance as a basis for benchmarking a parallel program (expert level) (see also PE2-E Measuring System Performance)

Ability to benchmark a parallel program (expert level) (see also PE3-I Benchmarking)

Ability to tune a parallel program from the outside via runtime options (expert level) (see also PE4.1-E Tuning without Building a Parallel Program)

Ability to apply the workflow for tuning (expert level) (see also PE5-B Optimization Cycle)

Ability to use a workload manager to allocate HPC resources for running a parallel program interactively (intermediate level)

Ability to run parallel programs in an HPC environment (basic level)

Ability to use a workload manager like SLURM or TORQUE to allocate HPC resources (e.g. CPUs) and to submit a batch job (basic level)

USE3-E Building of Parallel Programs

Relevant for: Builder and Developer

Description:

Ability to run parallel programs in an HPC environment (expert level) (see also USE2-E Running of Parallel Programs)

Ability to use a linker and to assess the effects of advanced linker specific options and environment variables (e.g. LIBRARY_PATH) (expert level)

Ability to configure the relevant settings (e.g. by setting compiler and linker options), which determine how the application ought to be build with regard to advanced parallelization technique(s) used (e.g. CUDA) (expert level)

Ability to use the profile guided optimization (PGO) technique (expert level) (see also PE4.2-I Tuning without Modifying the Source Code)

Ability to use software building environments like Scons, Waf, make, Autotools, CMake (expert level) (see also SE3.5-E Deployment Management)

Ability to use a compiler and to asses the effects of optimization switches available for compilers commercially available (e.g. PGI, NAG) (intermediate level)

Ability to use efficient open source libraries (e.g. OpenBLAS, FFTW) or highly optimized vendor libraries (e.g. Intel-MKL, IBM-ESSL) (intermediate level)

Ability to configure the relevant settings (e.g. by setting compiler and linker options), which determine how the application ought to be build with regard to the parallelization technique(s) used (e.g. OpenACC, C++ AMP) (intermediate level)

Ability to build parallel programs, e.g. via open sources packages (basic level)

Ability to use a compiler and to asses the effects of optimization switches available for the relevant compilers (e.g. GNU, Intel) (basic level)

Ability to use a linker and to assess the effects of linker specific options and environment variables (e.g. -L and LIBRARY_PATH, -rpath and LD_RUN_PATH) (basic level)

Ability to configure the relevant settings (e.g. by setting compiler and linker options), which determine how the application ought to be build with regard to the parallelization technique(s) used (e.g. OpenMP, MPI) (basic level)

USE4-E Developing Parallel Programs
Relevant for: Developer
Description:
Ability to build parallel programs (expert level) (see also USE3-E Building of Parallel Programs)

Ability to develop parallel software (expert level) (see also SE-E Software Engineering for HPC)

PE-E Performance Engineering
Relevant for: Tester, Builder, and Developer
Description:
Ability to use systematic approaches (e.g. benchmarking and tuning, cost models) to meet performance requirements in a cost-effective way, i.e. by reducing the runtimes of parallel programs and using the resources of the HPC system appropriately for that purpose (according to the respective level)

PE1-E Cost Awareness
Relevant for: Tester, Builder, and Developer
Description:
Ability to assess the costs related to the runtimes of parallel programs against the background of cost models (expert level) (see also K5-E Modeling Costs)

Ability to assess the ratio of personnel costs to resource costs against the background of break-even considerations and time to solution constraints (expert level)

PE2-E Measuring System Performance
Relevant for: Tester, Builder, and Developer
Description:
Ability to measure the system performance by the help of standard tools and by profiling in order to assess the runtime behavior of parallel programs (according to the respective level)

PE2.1-I Using Standard Tools to Measure System Performance
Relevant for: Tester, Builder, and Developer
Short Background: Tools used in Linux-based environemnts like htop, iostat, and perf are sketched. (intermediate level) This includes information about utilization of resources like CPU as well as elapsed runtimes of a program, its unshared and shared memory usage, input and output statistics for devices and file systems, and page faults, with tools like /usr/bin/time, ps, top, and vmstat in Linux-based environments. (basic level)
Description:
Ability to use standard tools of the operating system to get information about the behavior of parallel programs in terms of their resource utilization (basic level)

PE2.2-E Profiling
Relevant for: Developer

Short Background: Profiling is explained for the CPU level, where it can be supported by hardware performance counters and by sampling techniques. (basic level) Sampling is used to see, by examining the program counter, what routines and source code lines of a program are responsible for which portions of the total runtime. (basic level) Automatically adding trace code to a parallel program by so-called instrumentation to record its execution in a strict chronology is explained and the difference to profiling is emphasized. (basic level) Similar techniques are explained for profiling the network level (e.g. based on InfiniBand counters and I/O server states). (basic level)
Description:
Ability to detect performance issues and bottlenecks caused, for example, by inefficient programming, memory accesses, I/O operations, cache-misses, page-faults, and parallelization overheads (expert level) (see also K3.3-I Parallelization Overheads)
Ability to use advanced performance analysis tools like Vampir (expert level)
Ability to use the standard MPI profiling interface (PMPI) to control the built-in performance analysis functionality in MPI (expert level)
Ability to get the base data for tuning the performance of parallel programs by profiling (intermediate level)
Ability to assess how different views of the profiling data (e.g. timeline graphs and communication matrices to illustrate the traffic between processes) can give insights in the runtime behavior of the program (intermediate level)
Ability to use performance analysis tools like ScoreP, Scalasca (intermediate level)
Ability to use environment variables like $I_MPI_STATS to control the built-in performance analysis functionality in MPI (basic level)

PE3-I Benchmarking
Relevant for: Tester, Builder, and Developer
Description:
Ability to assess speedups and efficiencies as the key measures for benchmarks of a parallel program (intermediate level) (see also K2.1-I Performance Frontiers)
Ability to differentiate between strong and weak scaling (intermediate level)
Ability to assess the performance characteristics of parallel programs with regard to CPU usage, memory accesses (e.g. latencies for random access, cache sizes, strided access patterns, and bandwidth), I/O operations (e.g. record length, IOPs, latency, bandwidth, throughput, and multi-stream processing), and communication (message sizes, network bandwidth and latency) (intermediate level)
Ability to benchmark the runtime behavior of parallel programs, performing controlled experiments by providing varying HPC resources (e.g. 1, 2, 4, 8, ... cores on shared memory systems or 1, 2, 4, 8, ... nodes on distributed systems for the benchmarks) (basic level)

PE4-E Tuning
Relevant for: Tester, Builder, and Developer
Description:
Ability to tune a parallel program in order to achieve better runtimes and to optimize the usage of the HPC resources (according to the respective level)

PE4.1-E Tuning without Building a Parallel Program
Relevant for: Tester, Builder, and Developer
Description:
Ability to select appropriate tasks sizes (big vs. small) that may have positive

performance impacts on the workflow, and to run several (smaller) tasks by the help of job chaining (expert level) (see also USE1.2-E Using Shell Scripts)
Ability to use mapping of processes to nodes, pinning of processes/threads to CPUs or cores, and setting memory affinities to NUMA nodes in order to speed up a parallel program (intermediate level)
Ability to speed up program execution by setting appropriate runtime options (e.g. for MPI and OpenMP) (intermediate level)

PE4.2-I Tuning without Modifying the Source Code
Relevant for: Builder and Developer
Description:
Ability to speed up program execution by setting appropriate compiler/linker options for the PGO workflow (intermediate level)
Ability to speed up program execution by using optimized libraries and setting appropriate compiler/linker options (basic level)
Ability to speed up program execution by setting package specific options (e.g. selected by environment variables and command line arguments) (basic level)

PE4.3-E Tuning via Reprogramming
Relevant for: Developer
Short Background: The potential for tuning via reprogramming exists on the hardware as well as on the software level. At the software level, performance improvements are achievable by using more efficient algorithms. This is explained by the help of popular practice-relevant examples. (expert level)
Description:
Ability to run parallel programs in an HPC environment (expert level) (see also USE2-E Running of Parallel Programs)
Ability to reprogram appropriate parallel code for improved performance on the processing element level e.g. by using functional units (for executing fused multiply-add instructions and variants thereof), by using vectorization techniques with SIMD instructions, etc. (expert level)
Ability to assess how appropriate computationally intensive functions (which have been identified earlier by profiling the parallel program) can be ported to many core archictures like GPUs to achieve further speedups (expert level)

PE5-B Optimization Cycle
Relevant for: Tester, Builder, and Developer
Short Background: The workflow is represented by an optimization cycle with the steps benchmarking, gathering system performance data (e.g. via profiling), analyzing, and tuning.
Description:
Ability to apply the full workflow for tuning a parallel program (according to the respective level)

SE-E Software Engineering for HPC
Relevant for: Developer
Short Background: The user learns how to use practices and methods from software engineering that are especially important for high performance engineering.
Description:
Ability to understand software engineering methods and practices especially in the context of high performance computing (according to the respective level)

Ability to develop parallel programs and to apply software engineering methods and best practices (according to the respective level)

SE1-E Programming Concepts for HPC
Relevant for: Developer
Description:
Ability to develop programs for HPC (according to the respective level)
Ability to develop parallel programs for shared memory systems as well as for message passing systems (according to the respective level)
Ability to understand efficient algorithms and data structures (according to the respective level)
Ability to assess the efficiency and suitability of algorithms and data structures for the respective application (according to the respective level)

SE1.1-E Programming Languages
Relevant for: Developer
Short Background: The user learns how to complete programming tasks and gets a short overview of machine- and assembly-languages toward so-called high-level programming languages. The focus lies on the programming languages that are in widespread use within the HPC community.
Description:
Ability to use program language extensions used in HPC environments, such as HPX, Cilk (expert level)
Ability to program in advanced languages typically used in HPC environments, such as C++, Python (intermediate level)
Ability to use interoperability between languages, for example by calling C or C++ from FORTRAN and vice versa (intermediate level)
Ability to program in languages typically used in HPC environments, such as C, FORTRAN (basic level)

SE1.2-E Parallel Programming
Relevant for: Developer
Short Background: Parallel programming of shared memory systems and message passing systems as well as load balancing is addressed.
Description:
Ability to assess the parallel nature of algorithms (according to the respective level)
Ability to develop parallel programs (according to the respective level)

SE1.2.1-I Parallel Algorithms
Relevant for: Developer
Description:
Ability to assess that there are algorithms having a so-called sequential nature that have been notoriously difficult to parallelize, for example alpha-beta game-tree search (intermediate level)
Ability to understand that some algorithms are embarrassingly (i.e. trivially) parallelizable while their parallelization will vary from easy to hard in practice (basic level)
Ability to determine the computational complexity of algorithms (basic level)

SE1.2.2-E Programming Shared Memory Systems
Relevant for: Developer

Short Background: The parallel concepts of threads and processes are introduced and their impacts on performance are outlined.
Description:
Ability to assess concepts like software pipelining, e.g. to optimize loops by out-of-order execution (expert level)
Ability to assess the applicability of parallel language extensions like CUDA as well as their interoperability (e.g. combining OpenACC and CUDA) (expert level)
Ability to assess parallel concepts typically used for shared memory systems, e.g. to exploit temporal locality by data reuse with an efficient utilization of the memory hierarchy (intermediate level)
Ability to assess concepts like software pipelining, and vectorization principles (intermediate level)
Ability to assess the influence of control dependencies by jumps, branches, and function calls, e.g. on pipeline filling (intermediate level)
Ability to assess the applicability of parallel language extensions like OpenACC, and C++ AMP (intermediate level)
Ability to understand race conditions and to use synchronization mechanisms to avoid them (basic level)
Ability to understand the problems that may result from erroneous use of synchronization mechanisms (e.g. deadlocks) (basic level)
Ability to assess data dependency situations, i.e. an instruction reading the data written by a preceding instruction in the source code, and anti dependencies, i.e. an instruction having to read data before a succeeding instruction overwrites it, and output dependencies, i.e. instructions writing to the same memory location (basic level)
Ability to use data parallelism, e.g. applying parallel streams of identical instructions to different elements of appropriate data structures such as arrays (basic level)
Ability to understand the concept of functional parallelism, i.e. executing a set of distinct functions possibly using the same data (basic level)
Ability to assess the applicability of parallel language extensions like OpenMP (basic level)

SE1.2.3-E Programming Message Passing Systems
Relevant for: Developer
Description:
Ability to assess the impact of communication and synchronization on the performance of a parallel program (expert level) (see also K3.3-I Parallelization Overheads)
Ability to understand the concept of overlay networks (expert level)
Ability to understand the various communication modes (e.g. blocking vs. non-blocking, point-to-point vs. collective) (basic level)
Ability to develop programs using MPI as the de-facto standard for parallelizing programs in distributed environments like HPC cluster systems (basic level)
Ability to understand how race conditions and deadlocks may occur in MPI parallelized programs and how they can be avoided, namely by reordering send and receive operations or using non-blocking communication combined with waiting for completion of the communication operations concerned (basic level)

SE1.2.4-E Load Balancing
Relevant for: Developer
Description:
Ability to apply domain decomposition strategies (expert level) (see also K3.4-E Domain Decomposition)
Ability to apply more sophisticated approaches e.g. based on tree structures like divide-and-conquer or work-stealing to achieve an appropriate distribution of the workloads across the multiple computing resources of the HPC system (intermediate level)
Ability to apply simple scheduling algorithms like task farming to achieve an appropriate distribution of the workloads across the multiple computing resources of the HPC system (basic level)

SE1.2.5-I I/O Programming
Relevant for: Developer
Description:
Ability to assess general concepts of HPC I/O systems (e.g. parallel file systems) and how to map the data model to the storage system, e.g. by using appropriate I/O libraries and middleware architectures (intermediate level) (see also K1.3-I I/O Architectures)

SE1.3-B Efficient Algorithms and Data Structures
Relevant for: Developer
Description:
Ability to assess the efficiency of algorithms and data structures, especially with respect to their suitability for typical (scientific) (parallel) programs, e.g. by the help of popular practice-relevant examples (basic level)

SE2-E Programming Best Practices for HPC
Relevant for: Developer
Short Background: This skill provides knowledge about software development best practices that will help scientists to develop high-quality scientific software.
Description:
Ability to understand the best practices from software engineering regarding programming (according to the respective level)
Ability to apply programming best practices in order to develop robust and maintainable programs (according to the respective level)

SE2.1-B Integrated Development Environments
Relevant for: Developer
Description:
Ability to configure and use integrated development environments (IDEs) like Eclipse, e.g. to seamlessly perform the typical development cycle with the steps edit, build (compile and link), and test (basic level)

SE2.2-I Debugging
Relevant for: Developer
Description:
Ability to use sophisticated debuggers such as DDT and TotalView (intermediate level)
Ability to debug a program using simple techniques such as inserting debugging output statements into the source code, e.g. using printf – also against the background of potential problems with the ordering of the (stdout) output that may exist in parallel environments like MPI (basic level)

Ability to understand the common concepts and workflows when using a debugger (commands like step into, step over, step out, breakpoints) (basic level)
Ability to use sophisticated debuggers such as GDB (basic level)

SE2.3-E Programming Idioms
Relevant for: Developer
Short Background: This skill conveys programming idioms in general and for specific programming languages in order to help developers to solve recurring programming problems.
Description:
Ability to recognize where programming idioms are violated and to refactor the code to comply to a specific programming idom (expert level)
Ability to apply programming idioms for a specific programming language, e.g. FORTRAN, Python, C, C++ (intermediate level)
Ability to understand programming idioms for a specific programming language, e.g. FORTRAN, Python, C, C++ (basic level)

SE2.4-E Logging
Relevant for: Developer
Short Background: Logging is necessary in order to comprehend when, where, and why an error occurs during the execution. Parallel programs are prone to failures and errors during operation. Knowledge about logging concepts, the ability to apply them appropriately, and to purposefully analyze the log files is therefore essential in the context of high performance computing.
Description:
Ability to develop, maintain, and document a consistent logging concept for a program (expert level)
Ability to implement a logging concept for a program in a specific programming language, e.g. FORTRAN, C, C++ (expert level)
Ability to understand logging demands and challenges especially for distributed systems (intermediate level)
Ability to choose the most appropriate log format for the context (intermediate level)
Ability to apply structured logging and text logging (intermediate level)
Ability to understand logging in general like log levels etc. (e.g. ERROR, WARN, INFO, DEBUG, TRACE) (basic level)
Ability to understand different logging formats (basic level)
Ability to choose appropriate information that should be logged e.g. timestamp, pid, thread, level, loggername) in order to be able to identify the problem (basic level)
Ability to differentiate between structured logging and text logging (basic level)
Ability to understand logging implementations/libraries for a specific programming language like FORTRAN, C, C++ (basic level)

SE2.5-I Exception Handling
Relevant for: Developer
Short Background: The skill conveys general concepts about exception handling, how exception handling can be implemented in a specific programming language and how a consistent exception handling policy can be defined and thoroughly followed during implementation.

Description:
Ability to use best practices for exception handling (intermediate level)
Ability to understand how exception handling is supported in a specific programming language, e.g. FORTRAN, C (e.g. <errno.h>), C++ (i.e. try, catch, throw) (intermediate level)
Ability to apply appropriate exception handling in a specific programming language (intermediate level)
Ability to understand the differences between the terms "mistake", "fault", "failure", and "error" (basic level)
Ability to understand exception handling concepts in general (e.g. Errors vs. Exceptions) (basic level)
Ability to understand why it helps to write software that is robust (basic level)

SE3-E Software Configuration Management
Relevant for: Developer
Description:
Ability to apply steps of SCM in a HPC project (expert level)
Ability to understand the purpose and importance of software configuration management, especially in the context of high performance computing (basic level)
Ability to understand the basic concepts, terms and processes of SCM (basic level)

SE3.1-B Terminology (IEEE Standard)
Relevant for: Developer
Description:
Ability to understand the basics of SCM from the IEEE standards (e.g. IEEE-Standard 729-1983, IEEE 828: Software Configuration Management Plans, IEEE 1042: Guide to Software Configuration Management (basic level)
Ability to understand terms like Configuration Item, Baseline, SCM Directories, Version, Revision, Release (basic level)

SE3.2-E Version Control
Relevant for: Developer
Short Background: This skill covers how to apply version and configuration management to the development of (parallel) programs in order to track and control changes in the sources and how to establish and maintain consistency of the program or software system throughout its life, and facilitate cooperative development. (basic level) Systems like Revision Control System (RCS), Subversion (SVN), and GIT are presented. (basic level)
Description:
Ability to apply a specific workflow, such as Feature Branch Workflow, Gitflow Workflow, Centralized Workflow, Forking Worfklow (expert level)
Ability to apply advanced git concepts and commands (expert level)
Ability to apply Git as a version control system (intermediate level)
Ability to apply SVN as a version control system (intermediate level)
Ability to understand advanced git concepts, such as pull requests, branches, tags, submodules etc. (intermediate level)
Ability to understand different types of workflows, such as Feature Branch Workflow, Gitflow Workflow, Centralized Workflow, Forking Worfklow (intermediate level)
Ability to understand the basics of version control systems, e.g. what is version control (basic level)
Ability to understand the benefits of using version control for software development especially in a team; what is branching and merging (basic level)

Ability to assess the difference between distributed and centralized version control systems (basic level)
Ability to understand basic Git concepts (basic level)
Ability to understand basic SVN concepts (basic level)
Ability to use basic git commands such as git add, git commit, git pull, git push (basic level)
Ability to use SVN commands (basic level)
Ability to resolve merge conflicts (basic level)

SE3.3-I Issue Tracking and Bug Tracking
Relevant for: Developer
Description:
Ability to apply issue tracking in order to manage tasks, bug reports, and other issues occuring during development and enabling task assignment in the team (according to the respective level)
Ability to apply different issue tracking systems, like Jira or Redmine, to the development project (according to the respective level)
Ability to define a consistent workflow in the development team for issue tracking (according to the respective level)
Ability to understand concepts of issue/bug tracking systems and their basic concepts like task, sub-task, new feature, story, release planning, sprint planning in order to structure and organize the development process (e.g. assigning tasks to developers, reporting bugs, writing user stories, managing the stages of an issue (to do, in progress, in review, done) etc.) (according to the respective level)
Ability to understand different issue tracking systems, like Jira or Redmine (according to the respective level)

SE3.4-E Release Management
Relevant for: Developer
Short Background: The benefits of release management are explained. Moreover, it is covered how software releases are managed according to a well-defined and consistent process.
Description:
Ability to apply frameworks of release planning and management like SCRUM release planning or ITIL (expert level)
Ability to classify releases according to release categories (e.g. major, minor, emergency fix) (intermediate level)
Ability to plan and manage releases of scientific software and to document the release including the release notes (intermediate level)
Ability to apply best practices to make a release identifiable via version numbers using appropriate version numbering scheme (e.g. using the version control system) (intermediate level)
Ability to find the best release management process for the team (e.g. depending on team size etc.) (intermediate level)
Ability to understand the basics of release management and what the benefits are of applying a release management process in the context of high performance computing ”fault”, ”failure”, and ”error” (basic level)
Ability to understand the differences between Major Release, Minor Release, Emergency Fix (and potentially other types of releases) and what should be contained in each of them (basic level)
Ability to understand the tasks and steps of release management (basic level)

Ability to understand the steps of the deployment process of the release version and the required dependencies (basic level)
Knowledge of best practices of making releases identifiable via version numbers using appropriate version numbering scheme (e.g. using the version control system) (basic level)
Ability to understand the lifecycle of a release (including states such as stable, unstable) (basic level)
Knowledge of different frameworks of release planning and management, e.g. SCRUM release planning or ITIL (basic level)

SE3.5-E Deployment Management
Relevant for: Developer
Short Background: This skill conveys how dependencies are managed, how and why to setup different environments for development, testing, and production, how and why to automate the deployment process and the importance of preserving and documenting reproducible software stacks that can be used by other users/researchers in order to reliably reproduce the results.
Description:
Ability to setup the production, testing, and development environments (expert level)
Ability to clearly define and preserve reproducible software stacks to make computational results reproducible, e.g. by applying virtualization environments like VirtualBox, Docker, rkt, or BioContainers or tools for defining scientific workflows like Nextflow, or Singularity (expert level)
Ability to configure an environment for continuous integration, delivery, and deployment using a selected continuous integration tool like Jenkins, Buildbot or Travis-CI with basic processing steps like compiling and automated testing (expert level)
Ability to understand the basics of dependency management for different programming languages (intermediate level)
Ability to use advanced software building environments like Scons and Waf (intermediate level)
Ability to to undestand challenges for Portability e.g. for the source code of programs and job scripts to avoid typical compiler-, linker-, and MPI-issues. (intermediate level)
Ability to use a software build and installation framework like EasyBuild (intermediate level)
Ability to understand the basics of dependency management (basic level)
Ability to understand that different environments for testing, development, production, and staging are necessary (basic level)
Ability to understand the differences between different deployment environments, and what specific requirements are for each (basic level)
Ability to use software building environments like make, Autotools, CMake (basic level)
Ability to understand continuous integration, delivery, and deployment and the differences between them (basic level)
Ability to understand the benefits, drawbacks, and tradeoffs of continuous integration, delivery, and deployment (basic level)

SE4-I Agile Software Development
Relevant for: Developer
Short Background: Practices of agile software development are covered in order

to convey skills about collaborative, and self-organizing software development advocating adaptive planning, evolutionary development, and encouring rapid and flexible response to change. (basic level)

Description:
Ability to understand and apply agile development practices in the context of HPC (according to the respective level)

SE4.1-I Test-driven Development and Agile Testing

Relevant for: Developer
Description:
Ability to apply unit testing in a specific programming language using an appropriate unit testing framework, e.g. pfUnit for Fortran, glib testing framework for C (intermediate level)
Ability to develop (agile) testing strategies (intermediate level)
Ability to write different test types for the test pyramid (intermediate level)
Ability to understand the challenges of testing scientific applications (basic level)
Ability to understand test-driven and test-first concepts and understanding the benefits (basic level)
Ability to understand what constitutes a test strategy (basic level)
Ability to understand that there are different test types, e.g. given by the test pyramid (basic level)

SE4.2-I Extreme Programming

Relevant for: Developer
Description:
Ability to apply the principles in the context of an HPC project (intermediate level)
Ability to understand the principles of extreme programming and when to apply it (basic level)

SE4.3-I SCRUM

Relevant for: Developer
Description:
Ability to apply practices of SCRUM (intermediate level)
Knowledge of the concepts of SCRUM, e.g. Sprint, Backlog, Planning, Daily meetings/Stand up meeting, project velocity (basic level)

SE5-E Software Quality

Relevant for: Developer
Short Background: The benefits of software quality (functional and non-functional) are presented. Additionally, codings standards are covered helping developers to increase the quality of the code for better maintainability.

Description:
Ability to assess and improve the quality of the software especially with respect to functionality and maintainability (according to the respective level)

Ability to have the awareness of technical debt during software development and how to pay technical debt (according to the respective level)

SE5.1-E Coding Standards

Relevant for: Developer
Description:
Ability to define and establish coding standards and conventions in a project (expert level)

Ability to adhere to coding standards and conventions in a project (intermediate level)
Ability to understand common coding conventions, e.g. in order to ensure portability of the code (that are specific to a programming language, e.g. MISRA C) (basic level)

SE5.2-E Code Quality
Relevant for: Developer
Short Background: The user learns how to use practices and methods from software engineering that are especially important for high performance engineering.
Description:
Ability to understand software engineering methods and practices especially in the context of high performance computing (expert level)
Ability to develop parallel programs and to apply software engineering methods and best practices (expert level)
Ability to assess code quality using different metrics, e.g. length of functions, length of files, lines of code, complexity metrics, code coverage (intermediate level)
Ability to use static code analysis tools in order to calculate the metrics (e.g. http://cppcheck.sourceforge.net/) (intermediate level)
Ability to identify bad code structures (known as bad smells) in order to assess the quality of the code design (intermediate level)
Ability to understand different code metrics that assess code quality (basic level)

SE5.3-I Refactoring
Relevant for: Developer
Description:
Ability to apply common code refactorings in order to improve code quality, such as extract method, extract class, rename class and when it is suitable to apply which refactoring (intermediate level)
Ability to apply refactoring that are specific to programming languages (e.g. Fortran) (intermediate level)
Ability to understand different types of refactorings that help to improve the quality of the code and the design (basic level)

SE5.4-E Code Reviews
Relevant for: Developer
Description:
Ability to use a review system like Gerrit to organize the code reviews (expert level)
Ability to document code review results and resulting tasks in an issue tracking system (for example Jira) (expert level) (see also SE3.3-I Issue Tracking and Bug Tracking)
Ability to define checklists for code reviews (intermediate level)
Ability to conduct code reviews in pairs or in a team (intermediate level)
Ability to understand modern code reviews (basic level)
Ability to understand the single steps of a code review (basic level)

SE6-E Software Design and Software Architecture
Relevant for: Developer
Description:
Ability to design and develop programs on a higher level of abstraction by ap-

plying software architecture and object-oriented design (according to the respective level)

SE6.1-I Requirements Elicitation and Analysis
Relevant for: Developer
Short Background: Requirements elicitation, analysis and documentation are important, but often neglected, stages in developing scientific computing software. That is why the user will learn how to follow a systematic approach in order to appropriately collect and analyse requirements for the application and how to capture assumptions and constraints. Following this approach will help the user to significantly improve quality of scientific software.
Description:
Ability to identify functional and non-functional requirements of the software to be implemented (intermediate level)
Ability to capture a common terminology, constraints, and assumptions (intermediate level)
Ability to evaluate the identified requirements against the background of Software Quality (intermediate level) (see also **??**)
Ability to use the identified requirements to validate them by tests (intermediate level) (see also SE4.1-I Test-driven Development and Agile Testing)
Ability to use a predefined template in order to capture the requirements (intermediate level) (see also SE7.1-I Requirements Documentation)
Ability to understand the role of requirements elicitation and analysis during software development (basic level)

SE6.2-E Object-Oriented Design
Relevant for: Developer
Description:
Ability to apply design patterns for HPC (expert level)
Ability to understand design patterns for HPC (intermediate level)
Ability to write modular, reusable code by applying software design principles like Separation of concerns, loose coupling, information hiding, DRY, KISS etc. (following best practices like Clean Code by Robert C. Martin) (intermediate level)
Ability to apply object-oriented design and programming to scientific applications (intermediate level)
Ability to understand and use the main concepts of object-orientation (classes, interfaces, polymorphism) to design and implement a program (basic level)

SE6.3-E Software Architecture
Relevant for: Developer
Description:
Ability to design the application as a plugin architecture so that the functionality can be extended more easily (expert level)
Ability to design the software architecture of the system based on software architecture patterns (expert level)
Ability to develop an HPC application according to a reference architecture and adhering to the standards/guidelines given by the reference architecture (expert level)
Ability to understand component-based software architecture for scientific applications and ability to apply component-based architecture design (expert level)

Ability to understand architecture bad smells and ability to identify sub-optimal design decisions in architecture designs (e.g. sub-optimal communication between components) (expert level)
Ability to describe the software architecture of a scientific applications, what the most critical architecture decisions are, what the main structures of the system are, what the interfaces of the systems are, how they are designed and how components communicate with each other (intermediate level)
Ability to understand which software architecture patterns are typical for a specific type of scientific application and which aspects (e.g. quality attributes) they address (intermediate level)
Ability to apply design patterns to HPC, e.g. patterns for coding of parallel algorithms and their mapping to various architectures (intermediate level)
Ability to understand reference architectures for scientific applications (intermediate level)
Ability to understand architectural tactics for performance, scalability etc. (intermediate level)
Ability to apply software architecture principles during architecture design (intermediate level)
Ability to understand the characteristics and the architectural challenges of data-intensive and compute-intensive software systems and how they can be appropriately addressed (intermediate level)
Ability to understand the importance and impact of software architecture during software development (basic level)
Ability to understand software architecture principles (basic level)

SE7-E Documentation
Relevant for: Developer
Description:
Ability to appropriately document the entire software system (according to the respective level)
Ability to to provide a user documentation (according to the respective level)
Ability to provide a documentation for developers (e.g. describing the software architecture, for extending the software etc.) (according to the respective level)

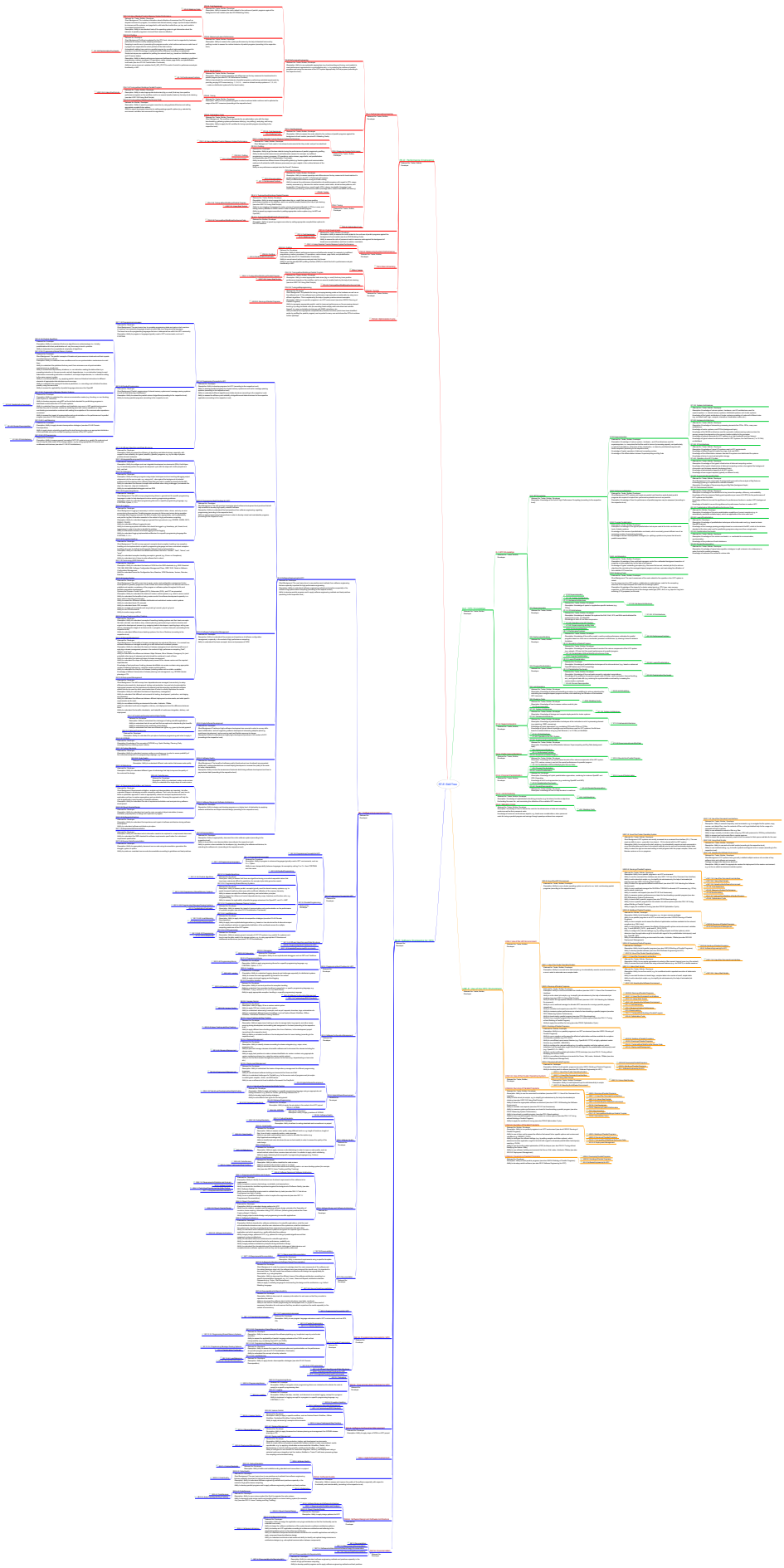
SE7.1-I Requirements Documentation
Relevant for: Developer
Description:
Ability to document requirements using a specified template (intermediate level)
Ability to understand which information needs to be captured in a requirements document (basic level)
Ability to understand the IEEE standard for software requirements specification for a structured requirement specification (basic level)

SE7.2-I Software Architecture and Software Design Documentation
Relevant for: Developer
Short Background: In order to preserve knowledge about the main components of the software and the related decisions about why the software have been designed this specific way, it is important to document them. This skill covers how software architecture and design can appropriately be documented, e.g. using templates.
Description:
Ability to document the different views of the software architecture according

to a specific documentation framework, e.g. 4+1 views, Views and Beyond, architecture decision frameworks (e.g. Taylor, Olaf Zimmermann) (intermediate level)
Ability to apply a modeling language for documenting the design and the architecture, e.g. Unified Modeling Language (intermediate level)

SE7.3-B Source Code Documentation
Relevant for: Developer
Description:
Ability to appropriately document source code using documentation generators like doxygen, pydoc, or sphinx (basic level)
Ability to produce a consistent source code documentation according to guidelines and best practices (basic level)

SE7.4-E Documentation for Reproducibility
Relevant for: Developer
Description:
Ability to understand software engineering methods and practices especially in the context of high performance computing (expert level)
Ability to develop parallel programs and to apply software engineering methods and best practices (expert level)
Ability to document all necessary information for end-users so that they are able to reproduce the results (intermediate level)
Ability to document the software stack, build instructions, input data, results etc. (intermediate level)
Ability to use tools for literate programming like activepapers knitr, or jupyter to document all necessary information for end-users so that they are able to reproduce the results especially in the context of concurrency (intermediate level)



HPC Skill Tree