E-Infrastructures
H2020-EINFRA-2014-2015

EINFRA-4-2014: Pan-European High Performance Computing Infrastructure and Services

PRACE-4IP

PRACE Fourth Implementation Phase Project

Grant Agreement Number: EINFRA-653838

D5.6
Best Practices for Prototype Planning and Evaluation

Final

Version: 1.0
Author(s): Michael Ott (BADW-LRZ), Jeanette Wilde (BADW-LRZ), Torsten Wilde (BADW-LRZ), Hayk Shoukourian (BADW-LRZ), Carlo Cavazzoni (CINECA), Radek Januszewski (PSNC), Giannis Koutsou (CYI)
Date: 11.04.2017

Project and Deliverable Information Sheet

PRACE Project	**Project Ref. №: EINFRA-653838**	
	Project Title: PRACE Fourth Implementation Phase Project	
	Project Web Site: http://www.prace-project.eu	
	Deliverable ID: **< D5.6>**	
	Deliverable Nature: Report	
	Dissemination Level: PU*	**Contractual Date of Delivery:** 30 / April / 2017
		Actual Date of Delivery: 30 / April / 2017
	EC Project Officer: Leonardo Flores Añover	

* - The dissemination level are indicated as follows: **PU** – Public, **CO** – Confidential, only for members of the consortium (including the Commission Services) **CL** – Classified, as referred to in Commission Decision 2991/844/EC.

Document Control Sheet

Document	**Title: Best Practices for Prototype Planning and Evaluation**	
	ID: **D5.6**	
	Version: 1.0	**Status:** ***Final***
	Available at: http://www.prace-project.eu	
	Software Tool: Microsoft Word 2010	
	File(s): D5.6.docx	
Authorship	**Written by:**	Michael Ott (BADW-LRZ)
	Contributors:	Carlo Cavazzoni (CINECA) Radek Januszewski (PSNC) Giannis Koutsou (CYI) Hayk Shourkourian (BADW-LRZ) Jeanette Wilde (BADW-LRZ) Torsten Wilde (BADW-LRZ)
	Reviewed by:	Thomas Eickermann (JUELICH) Evangelos Floros (GRNET)
	Approved by:	MB/TB

Document Status Sheet

Version	Date	Status	Comments
0.1	25/January/2017	Draft	Structure
0.2	29/March/2017	Draft	Input from Carlo, Hayk, and Radek
0.3	30/March/2017	Draft	Summary, Additional input by Carlo
0.4	03/April/2017	Draft	Formatting, comments from Torsten, Input from Giannis
0.5	11/April/2017	Draft	Incorporated review recommendations
1.0	15/April/2017	Final version	Incorporated comments by Jeanette

Document Keywords

Keywords:	PRACE, HPC, Research Infrastructure, Prototyping

Table of Contents

List of Tables

References and Applicable Documents

[1] G.Erbacci, C. Cavazzoni. “Systems compliant with user requirements”. PRACE-PP Deliverable D7.2.

[2] H.-D. Steinhöfer. “Initial Risk Register”. PRACE-PP Deliverable D7.4.1.

[3] L. Johnsson, G. Netzer, E. Boyer, S. Graf, W. Homberg, G. Koutsou, J. Jakic, R. Januszewski, N. Puzovic, T. Roeblitz, O.-W. Saastad, B. Schembera, G. Schwarz, H. Shoukourian, V. Strumpen, S. Thiell, G.-C. deVerdiere, and T. Wilde. “Report on prototypes evaluation”. PRACE-1IP Deliverable D9.3.3. 2013.

[4] L. Johnsson, G. Netzer, E. Boyer, G. Koutsou, R. Januszewski, P. Carpenter, O.-W. Saastad, and T. Wilde. “Final Report on Prototypes Evaluation”. PRACE-1IP Deliverable D9.3.4. 2013.

[5] R. Januszewski, O.-P. Lehto, C. Cavazzoni, T. Wilde, J. Wilde. “Final Prototyping and Technical Evaluation Summary”. PRACE-2IP Deliverable D11.1.3. 2014.

[6] J. Eriksson, R. Januszewski, O.-P. Lehto, C. Cavazzoni, T. Wilde, J. Wilde. “[1] R. Januszewski, O.-P. Lehto, C. Cavazzoni, T. Wilde, J. Wilde. “Final Prototyping and Technical Evaluation Summary”. PRACE-2IP Deliverable D11.1.3. 2014.”. PRACE-2IP Deliverable D11.2. 2014.

[7] G. Koutsou, T. Wilde, J. Wilde, J. Follows. “User-side Prototype Evaluation”. PRACE-2IP Deliverable D11.3. 2014.

[8] S. Bernardi, P. Segers, I. Yenes. “First Report on the joint PCP Pilot (Phase I)”. PRACE-3IP Deliverable D2.1.2. 2014.

[9] Mankins, John C. "Technology readiness levels". NASA White Paper. 1995

List of Acronyms and Abbreviations

aisbl	Association International Sans But Lucratif (legal form of the PRACE-RI)
EC	European Commission
GPU	Graphic Processing Unit
HPC	High Performance Computing; Computing at a high performance level at any given time; often used synonym with Supercomputing
MPI	Message Passing Interface
PRACE	Partnership for Advanced Computing in Europe; Project Acronym
TCO	Total Cost of Ownership. Includes recurring costs (e.g. personnel, power, cooling, maintenance) in addition to the purchase cost.
TDP	Thermal Design Power

List of Project Partner Acronyms

BADW-LRZ	Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Germany (3rd Party to GCS)
CaSToRC	Computation-based Science and Technology Research Center, Cyprus
CINECA	CINECA Consorzio Interuniversitario, Italy
GCS	Gauss Centre for Supercomputing e.V.
GRNET	Greek Research and Technology Network, Greece
JUELICH	Forschungszentrum Juelich GmbH, Germany
PRACE	Partnership for Advanced Computing in Europe aisbl, Belgium
PSNC	Poznan Supercomputing and Networking Center, Poland

Executive Summary

Prototyping is an important tool in High Performance Computing to test new concepts, evaluate novel technologies, and assess trends in hardware and software development. Proper planning, execution, and evaluation of such prototyping projects is key to their success. Building on experience from previous PRACE and other FP7 and H2020 prototyping projects, this best practice guide aims at providing a comprehensive guide for the individual phases of HPC prototyping projects.

1 Introduction

Prototyping is an important activity to evaluate new concepts and technologies for exa-scale computing. It was one of the main activities during the PRACE-PP [1,2], PRACE-1IP [3,4] and PRACE-2IP [5-7] as well as PRACE-3IP [8] projects. These activities were moved into a separate line item by the EC and prototype technology projects such as Mont-Blanc and DEEP were funded. Within PRACE-3IP three prototype systems are being procured via a pre-commercial procurement (PCP) scheme.

This deliverable summarizes the experiences gathered during the PRACE projects related to the prototyping process. It is based on personal experience of the authors during these projects, input from other WP5 partners, and PRACE deliverables on the subject [1-8] . It will provide high-level guidance and best practice recommendation for the continuation of HPC prototyping in EC funded efforts.

Chapter 2 will discuss the process of prototype planning covering the concept phase and the project planning. Chapter 3 will provide recommendations related to the execution of the prototyping effort. This covers the prototype deployment as well as prototype operation. Chapter 4 will provide guidance related to the evaluation of a prototype system. It highlights the evaluation criteria and discusses important considerations related to the user perspective of prototype evaluation. Finally, Chapter 5 will provide the summary for this deliverable.

The expected audience for this deliverable is European institutions and research consortia that are already actively involved or considering the use of IT technology prototypes to move European IT technology forward.

2 Planning an HPC Prototype

The activities in the planning phase of an HPC prototype can be grouped in two main categories. In the first group there are all activities related to the building of the prototype concept, like the analysis of the needs for innovations, limitations, and brainstorming. The second group includes all activities related to the prototype project plan, like task break down structure, activities GANTT and PERT, risk analysis, and other activities related to project design.

2.1 Prototype Concept

The idea of developing a prototype in HPC usually originates from a need that is not satisfied from the available solutions on the market or the lack of a functionality relevant for certain HPC workloads. Another important issue, that in our experience motivates a prototype, is the need to improve the efficiency of a computing system to lower the total cost of ownership of the infrastructure. Today, supercomputers are assembled from off-the-shelf components that are typically designed for a broader market and not specifically for HPC and HPC workloads. Therefore, there is room to optimize the level of integration and the working set point. HPC prototypes usually include both software and hardware components; the relevance of the two depends greatly on the nature of the prototype.

Once the need for a new solution has been identified, e.g. the lack of a solution for power aware scheduling to better manage system TCO, a make or buy decision needs to be made. PRACE projects have supported and experimented in both directions: in PRACE-1IP and PRACE-2IP, the prototype programs supported the “make” type of decision; whereas PRACE-3IP, with the PCP procurement, supported and verified the “buy” decision. As the PCP has its own specific knowledge base and PRACE deliverables, this deliverable will focus the discussion on the experiences for a “make” kind of prototyping project. However, this does not mean that vendors are not part of the prototype solution. On the contrary, vendors are usually an important partner in making a prototype and they need to find the prototype concept interesting and useful from an industrial point of view as well.

Once the make decision has been made, the brainstorming phase starts. Here a person (or persons) takes the responsibility of consulting all resources of the institution with competencies relevant for the prototype realization, surveying the market, collecting other similar experiences, and raising the interest of potential industrial partners. This brainstorming phase ends with a first draft proposal where the critical components and activities, including potential industrial partners, are identified. The draft should also contain a rough estimate of budget and scheduling. Furthermore, the approval from the institution is required together with the identification of potential funding schemes.

2.2 Project Plan

Before drafting a detailed project plan, a brief project charter with the concept, objectives, stakeholders, and a rough estimate of the budget and timeline is compiled to be used to present the idea to the institution decision board/panels to gain approvals and go ahead for the project.

2.2.1 *Drafting a Project Plan (timeline, guide execution and evaluation)*

The project plan for a prototype should in general follow the best practices of project management but should have a strong focus on the objectives and management of the risks. Moreover, with respect to other projects, the management of the execution phase should consider AGILE approaches meaning the project needs to adapt quickly to changes that may happen outside the project control (e.g. a delay in a critical component from a supplier or a change in the technical specs of a component). It might be obvious, but it is also very important for the success of the prototyping project to involve partners with the expertise needed for the success of the effort.

It is important to keep in mind that the success of a prototype is not only related to the technical outcome. While there is typically a high risk to fail, an even more important outcome is the knowledge gathered and the lessons learnt during the project execution. Consequently, attention should be paid to the curation of the information collected during the execution. Depending on the project sponsor, publishing one or more articles, books, or white papers about the prototype are a good opportunity to reach this goal.

Besides the above general remarks on project management guidelines, the following sections will give a more detailed description on individual topics that need to be covered by the project plan.

2.2.2 *Planning for the Right Technology Readiness Level*

The term "prototype" has a different meaning for different persons depending on their backgrounds. For a hardware developer it could mean a system that is being built from scratch and requires soldering to assemble and sophisticated tools and measurement equipment to bring to production. For a system administrator it could mean an off-the-shelf system that is being deployed to test the latest generation of (commercially available) hardware. Obviously, the skillset required to install and operate these two prototypes will vary greatly.

In order to create a common terminology, the system of Technology Readiness Levels (TRLs) was established. It allows for assessing the maturity of a particular technology and the corresponding risks associated with its development. Additionally, it enables a scale for the consistent comparison between different types of technologies [1]. The table below presents the descriptions of the nine level TRL scale (with TRL 9 being the highest level of technology maturation) and suggests a corresponding interpretation for HPC system hardware and system software.

Technology Readiness Level (TRL)	Definition	HPC System Hardware	HPC System Software
TRL 1	Basic principles observed and reported	Scientific research supporting the concepts and applications of the target hardware technology is in place.	Scientific research supporting the main properties of system software architecture and required formalizations is in place.
TRL 2	Technology concept and/or application formulated	The first manufacturing principles are identified – no formal analysis or experimental data/proof is available to underpin the approach.	Functional requirements identified. Basic concepts defined and essential primitives implemented. Primary tests might have been performed.

Technology Readiness Level (TRL)	Definition	HPC System Hardware	HPC System Software
TRL 3	Analytical and experimental critical function and/or characteristic proof-of-concept	Identification of target user requirements and market feasibility assessment. R&D on a laboratory scale (i.e. in an environment that does not necessarily correspond to the target operational environment to be encountered by the final high-end system). First simulation results validating the analytical predictions.	Development of the primary, non-integrated, kernel software functionalities to meet the predefined critical requirements.
TRL 4	Component and/or breadboard validation in laboratory environment	The first prototype version is built. The basic functionalities are validated on a laboratory scale. Requirements for the target operating environment(s) are defined.	Kernel software components integrated and validated within the test environment. Concepts and requirements of the interoperability between different components for the target operational environment defined.
TRL 5	Component and/or breadboard validation in relevant environment	The enhanced version of the prototype system is developed to meet the essential functionality requirements within the simulated relevant environment (i.e. within an environment that encompasses all the essential and critical aspects of the final target operational environment).	End-to-end software components implemented, interfaced with each other, and validated within the relevant environment.
TRL 6	System/subsystem model or prototype demonstration in a relevant environment	A high-grade prototype system is built. The overall functionalities demonstrated for the relevant operational environment(s). Predefined critical performance requirements are met.	Prototype implementation validated for realistic use cases. Partially integrated with the available hardware components. Initial documentation available.

Technology Readiness Level (TRL)	Definition	HPC System Hardware	HPC System Software
TRL 7	System prototype demonstration in relevant environment	A semi-final high-end system is built and operated in a relevant operational environment. The functionality for the target operational environment is demonstrated.	Prototype implementation meeting all the predefined key functionality requirements. Integrated with corresponding HW & SW components demonstrating the required operational feasibility.
TRL 8	Actual system completed and qualified through test and demonstration	The final high-end system is deployed in its final configuration. The outlined functionality of the system within the intended operational environment is demonstrated and validated through various tests and analysis.	The complete system software stack installed and fully integrated with the target operational system(s). The functionality has been thoroughly tested and demonstrated in simulated operational scenarios/environments.
TRL 9	Actual system proven through successful mission operations	The final high-end system is successfully operating within the targeted actual operational environment.	Software verification and validation completed. System successfully operated in the intended operational environment for the predefined acceptance period. All the corresponding documentation is completed and in place. Corresponding system software engineering support available.

Table 1: Description of the Technology Readiness Levels

Different from planning for other projects, in planning for a prototype it is important to make an accurate analysis on which TRL the prototype could achieve. Depending on the target TRL the different phases of the project execution may have different relevance. The TRL will also significantly influence the test and validation planning. An error in targeting the right TRL for the prototype will probably turn into issues and risks for the project.

2.2.3 *Planning for the right objectives and scope*

The purpose and the objectives of the prototype should be clearly stated and the extent to which each objective is linked to the HW/SW components needs to be precisely described. In our experience one should be aware of the fact that some objectives may be broader than a specific HW/SW implementation chosen for the prototype and the change in one of these components, even if critical for the risk management, does not change the objectives. For example, for the Eurora prototype at CINECA, the main objective was to validate a new hot water direct liquid cooling technology. The original plan was to equip the prototype with Intel Knight's Corner accelerators but due to a delay in the availability of components the prototype was initially deployed with Nvidia K20 general purpose GPUs which were later replaced with Intel Knight's Corner. This was managed as one of the identified risks but did not cause a change in the overall objectives.

2.2.4 *Planning for Relevant Metrics, Benchmarks and Targets.*

Together with the definition of the objectives, it is important for the success of the prototype to define a baseline and benchmark for the performance and the functionalities of the prototyped components/system. The baseline (e.g. PUE of setup XYZ, Flops/watt, etc.) is essential to understand if the prototype allows for progress beyond the state of the art. For example, as Flops/Watt was one element of the evaluation for the Eurora prototype, the Green500 list had been chosen as a reference for evaluating the prototype. While this is a rather simple metric one could also think of more sophisticated metrics that may prove more difficult to define and evaluate, like the usability of a system or a software.

2.2.5 *Planning for the Evaluation*

Almost as important as the planning before the project starts, is the evaluation of the outcome of the project. Already in the planning phase of the project the evaluation criteria should be defined. Chapter 4 will provide a comprehensive list of topics to cover during the evaluation phase.

2.2.6 *Planning for Gathering the Right Expertise*

It is important to decide who is involved in the prototype evaluation. In HPC, depending on the nature of the prototype, end users need to be involved to evaluate application performance. This was done in nearly all PRACE prototype evaluations. PRACE users as well as third party users were involved in the prototype experimentation/evaluation.

Having users testing the prototype could be an issue since production level user support can't be provided. Therefore, it is important to communicate limits and constraints to the user community to not raise unfunded expectation. This is of fundamental importance if meaningful information on their experience is to be gathered. It is also important to keep in mind that the user experience is an important source of information but can be biased.

2.2.7 *Planning for an Applications and Validation Suite*

For the success of the prototype, the project plan should foresee a set of applications to be used in the benchmark. If at the time of planning it is not possible to completely define this set, it is then important to define a methodology to identify the relevant applications.

Depending on the TRL of the prototype, it may not be possible to run end user applications. In this case, one possibility is to define a set of applications but, instead of planning to run the full applications of the benchmark set, analyse them and derive from them a set of synthetic kernels the applications are based on. The complexity of the kernels depends on the functionality of the prototype: they may consist only of a small set of elemental instructions and loops but could also be full application subroutines or libraries. In this case, besides the importance of choosing the right set of applications, it is more important to find a set of kernels that cover all relevant functionalities.

The prototyped component may not have a direct impact on the application performance (e.g. prototyping a new PDU). In this case, the applications may be used to build a mix that emulate a production workload to be used to run extensive tests that replicate production conditions.

2.2.8 *Planning for the Software Stack and Tools*

Depending on what kind of tests/validations planned, the software stack and tools need to be chosen accordingly. Again, the TRL of the prototype will influence the software stack. Different from a production system, quite often the software stack will include additional components to validate, monitor, and measure relevant prototype features and components.

In almost all cases, in our experience, a standard software stack needs to be modified to work with pre-release components (e.g. to support new drivers) and additional functionalities to compensate for prototype limitations (e.g. software that switches-off the prototype in case the prototype reaches critical conditions).

2.2.9 *Planning for the Facility*

In planning for the prototype, another important aspect, much more than for a product, is to consider all aspects related to the integration with the hosting facility. There are several things to take into account. First of all, one needs to consider if the prototype is compliant with security standards and if it holds all required certifications. In fact, it may happen that, depending on the TRL of the prototyped components, an engineering sample or the system as a whole is not certified for the electrical, mechanical, or electrical security standards. In this case, the prototype cannot be hosted together with other systems to avoid any risks.

Another reason why a prototype may require a dedicated room separated from other systems lays in the compatibility of the set point for ambient conditions. This may involve higher temperatures than what could be tolerated by production systems.

2.2.10 *Planning for the Integration with the Production Environment*

If the TRL of the prototype is high enough and/or the prototype objective is about a component that uses standard interfaces, the prototype may be compatible with a production environment. In this case, it could be valuable (like it has been in the context of the PRACE prototypes) to combine the prototype activities with the production environment and grant access to end users.

However, this depends on the maturity of the software stack of the prototype and operating it in a production environment may create a security risk to the production systems. If that is the case, the prototype should be operated on a private network.

2.2.11 *Planning Synergy with other Projects/Initiatives*

To reduce the risk of the project and better define the objectives of the project it is important to perform a survey about similar ongoing or planned initiatives. This can be done by searching publicly available information and by means of interviews of experts in the field (usually HPC innovators).

The outcome of this survey may include information about two classes of initiatives, one having overlapping funding agencies/sponsors and the other not.

Among the first there could be other prototype projects funded be EU whereas in the second there could be those funded e.g. in US or Japan.

For the first class, it is usually in the interest of the funding agency/sponsors to avoid overlaps and maximize synergies. Then it becomes important to establish a link with these other initiatives and agree on a given protocol to exchange information about the objectives and the status preserving the project independence.

For the second class, it is important to evaluate the scope of the competition between the two initiatives and between the funding agencies and act accordingly.

Usually, it is a win-win situation to exchange information about competitive activities and outcomes but it can be a risk for the prototype project to share information about the status update and on the TRL.

Note that there is always the risk of overlapping goal and scope with other prototype initiatives, especially if these are in preparation phase, or in an early stage, where no public information is available or disclosed.

3 Prototype Project Execution

After the design phase the prototype project comes to the execution phase where, depending on the quality of the design phase, one will experience most of the obstacles. This phase usually requires maximum flexibility as the prototypes by their nature are not standard deployments. Entering this stage puts us ahead of three classes of issues that can be summarized by a quote:

There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know.

Donald Rumsfeld

Known knowns

These are the items that were discussed during the design phase: what is the purpose of the prototype, how should it be constructed, what is required to build it, who will build what, what is the quality of the data we would like to collect from the prototype, and how should we do it. One needs to be ready to accept that some of the knowns might turn out to be not known and might move to the “unknown unknowns” class.

Known Unknowns

These, like the known knowns, should be covered during the design phase. This class is, in most cases, not very dangerous because some of them should be covered under “risks” and be addressed in a mitigation plan. Others will just be the goals of the prototype.

Unknown Unknowns

This class of issues is the most important as it covers all the unexpected conditions that may happen during the deployment phase. This class covers things that are assumed possible but turn out otherwise. Examples for this case may be things like: our datacentre policy/insurance/safety officer does not allow us to put non-certified equipment in the designated area; there is not enough power; and the form factor of a component of the prototype has been changed by the manufacturer and now does not fit in the chassis anymore.

This class will either influence the prototype assumptions, increase the cost of the prototype, or, if one is lucky, only cause some delays. However, independent of how well the prototype was planned problems of this class will happen.

3.1 Prototype Deployment

Depending on the type of prototype, the physical deployment may be the most problematic phase. By its nature, building a prototype usually involves skills and knowledge beyond the state of the art using exotic parts or technologies that were never used before. Usually this means that 3rd party contractors are involved in the procedure and this will probably result in delays or changes of the specifications. Good practice is to involve all contractors from the design phase and clearly state all assumptions and requirements, regardless how obvious they may seem. Usually the deployment can be split into phases: delivery; component installation and tests; integration of the components; and running the whole prototype.

3.1.1 *Delivery*

This phase, especially if the size of the prototype is significant, may be more or less problematic. Situations like “we have one door along the delivery path that is not big enough” and “there is a three-step stair, no lift, and the prototype weights 500kg” happen more often than one may think. Another case is the delivery schedule that may be influenced by external events. The more components from different vendors are required the more probable it is that the delivery will be delayed. The risk increases with the distance from the place where the manufacturing is happening. An example of these kinds of problems may be: “Yes, two months ago, when you asked us, we would deliver within two weeks but right now there is Chinese New Year and you have to wait six weeks” or “We will not deliver in July, everyone in Italy goes for vacation and our company is closed for three weeks”.

3.1.2 *Component installation and tests*

This phase, especially for the custom-built components, may also be a source of problems because usually the place where you install the prototype differs from the lab where the component was tested/built. A key factor for the success of this phase is good cooperation between the vendor and the integration teams especially when the integration requires some specific engineering skills (e.g. cooling). For the coordinator, acting as a man in the middle between the vendor and the integration team is not good practice, however, monitoring the process is crucial.

3.1.3 *Integration of the components.*

Depending on the nature of the prototype, this phase may be the most difficult part of the deployment phase. The problems that may arise may be as in previous examples of hardware or software nature. Often the combination of new drivers, required for the new hardware, and the requirement of a specific driver version for standard communication libraries (such as MPI) may create an issue that may be difficult to solve without significant effort.

Also during this stage, the instrumentation equipment and software is prepared. One must make sure that all required sensors are accessible, the resolution and polling frequency of all of them is good enough, and the readings are synchronized. For data collection and analysis, it is important to select software that supports open or industry standards especially if several prototypes should be compared.

3.1.4 *Running the whole prototype*

After successful installation, the prototype should be operational and ready to run as a single solution. While just starting the prototype that consists of well-tested components is not very problematic, achieving the desired performance levels is usually more difficult and is covered in the “prototype operation” section.

It may seem that the majority of the problems mentioned before could be foreseen and mitigated during the design phase. However, some of the activities probably will be done for the first time or at least for the first time for this kind of implementation by the majority of the parties involved. Hence, “unknown unknowns” will be abundant so a flexible adaptation plan is a key.

3.2 Prototype Operation

Once the prototype is successfully deployed, work on the prototype starts where not only the teams responsible for the installation and running are involved but also other teams that test different aspects of the installed system (performance, programming, software versions, etc.).

Depending on the technical maturity of the prototype and the risks of potential malfunction, a careful operation plan should be prepared. In the beginning of the operation, the prototype should be operated only when someone is working on it actively. Then, provided there are no accidents, the operation time might be extended to working hours when, in case of emergency, someone can act accordingly. Regardless of the timespan when the prototype is operational, the monitoring system should work 24/7 to gather any possible anomalies even if the prototype is not really doing any work. During the first stage, stress tests should be performed testing the safety and stability of the prototype by simulating extreme conditions or malfunctions of specific components. All the stress tests should be performed under active supervision of the responsible staff. Only after successful stress tests, the prototype should be allowed to run without supervision since long-term stability may also be a subject of tests.

Prototype operation will require interaction with the vendor and any 3rd parties involved in installation and design. Therefore, it is important to ensure that the contract also covers support during the operation phase as it is not uncommon that some adjustments to the system firmware or physical installation may be needed after first results from the prototype are gathered.

In many cases, there is more than one team interested in using the prototype so a proper feedback about problems, milestones, and deadlines is important to keep the flow of the work streamlined. In many cases starting work on a specific aspect of the prototype requires some preparation before the real work is started, therefore, all parties involved in the prototype operation should be informed of all activities not only maintenance shut downs and exclusive access schedules. It is important that all changes to the prototype be reported in advance as it may disrupt the work-plan of the other groups or even invalidate all data gathered thus far as the changes to the environment may make the results of the runs incompatible with ones done after.

4 Prototype Evaluation

As discussed in section 2, a prototyping project should have clearly defined targets and goals. To assess the outcome of the project, it is important to have a list and prioritization of evaluation criteria. The following subsections will give some guidance towards potential criteria and how to assess them. It is important to note that the evaluation of the project should not only take place at the project end but should be a continuous process during the execution phase to adapt and adjust to changing boundary conditions.

4.1 Identification and Prioritization of Features

An important prerequisite for assessing the success of a prototype is a clearly defined list of (planned) features and their priorities. Obviously, the intended features of the prototype will have to be defined in the planning phase but it is also possible (and typically happens) that features are added during the project execution, e.g. new technology becomes available or additional opportunities emerge with the use of the new technology. Other features may prove to be unfeasible to implement or might be dropped because they appear to be less important as the project progresses. In any case, the list of features needs to be kept up-to-date as the project moves along and each feature should have a priority assigned to it. During the evaluation phase of the project, each feature should be revisited, its importance re-evaluated, and finally checked, whether the final prototype provides it. At some point it is important to state to stakeholders that no further additions/changes may be made otherwise this process may never end; scope creep.

4.2 Assessment of Technology Readiness Level (Hardware & Software)

As outlined in chapter 2, the technology readiness level is an important attribute for a prototype system. It helps both users and operators to get an understanding of the maturity of the system. It is therefore important to assess the TRL of the final prototype.

4.3 Ease of Deployment and Operation

Prototyping, by definition, is different from a regular procurement. Typically, it will require more effort, skills, and manpower to deploy a prototype system, stabilize it, and operate it reliably. Obviously, this will largely depend on the technology readiness level (TRL) of the system but it is by far not the only determinant. Even mature technologies, and there might be quite a few of them even in a prototype, can be quite difficult to handle and require a lot of knowledge. Additionally, the TRL is subjective and depends significantly on the knowledge and skillset available within the organisation deploying the prototype. What is challenging to one might seem to be easy to another. It is therefore important to define the TRL during the planning phase specifically for the intended operational environment.

4.4 Software Stack and Tool Support

Even in a hardware prototyping project, software plays a key role for success. For the system operator, it will be important that a stable operating system with the necessary driver support is available. For application developers, the availability of compilers and debug tools will be key for software development on the prototype. For the users, it will be important that the runtime environment they need for their particular applications is available and runs without crashes.

4.5 Benchmark Applications

Particularly in High Performance Computing, performance is key to all sorts of hardware development. Hence, any prototyping project will also have to assess the performance of the prototype during the evaluation. To do this, a set of benchmark applications will be required that test important features of the hardware. Benchmarks could be implemented either as kernel benchmarks that only test a very specific feature or as (derived) application benchmarks that mimic the behaviour of actual user applications and hence test multiple features simultaneously. While kernel benchmarks have the advantage of being very specific to individual performance metrics, the benefit of application benchmarks is that they also give an idea about the general performance of a prototype system. Typically, one will use a mixture of both to benchmark the prototype.

4.6 Collect Lessons Learnt

The lessons learnt are probably among the most important benefits of a prototyping project. In any given project, there will typically be a gap between the project plan and the actual project execution. Particularly for long-term projects, there are so many external factors that have an influence on the project execution that it is hard to predict or foresee all of them or their potential impact. Experience from previous projects is very valuable for good project planning and for devising strategies to deal with unexpected events. It is therefore essential to collect the lessons learnt of a finished project for the planning of the next. Prototyping projects are, by definition, of higher risk than regular procurement projects. There is no shame if such a project deviates from the original planning or fails completely. The important thing here is to analyse potential failures or mistakes and share them in order to avoid similar problems in the next project.

4.7 KPI evaluation

Key Performance Indicators are a valuable tool to evaluate the success of a prototyping project. They do not only apply to the actual prototype but also on the project planning and execution. As such, they can prove very useful to assess the overall project performance. The KPIs on project planning and execution are generic and may apply to many different projects. Additionally, KPIs need to reflect the fact that a prototyping project can be of high risk and that the final prototype may not deliver all of the planned features. But as discussed in the previous section, it is the experience gained during the project that is of most value.

4.8 Evaluate Support of Technology Provider/Vendor

Like the lessons learnt discussed in the previous section, a prototyping project is also a valuable tool to assess technology providers and vendors. Prototyping projects require a high level of flexibility and a wide skill set. Hence, the performance of the vendor in the project and the way the vendor reacts to problems can provide good insight into the inner workings of the vendor and its capabilities. This can be useful for two purposes: for picking partners in the next prototyping project, but also, and more importantly, for selecting vendors in regular procurements.

4.9 User Perspective

Engaging users during prototype evaluation may be beneficial in assessing the prototype's readiness for production. Experienced users, namely users that develop or contribute to the scientific application they use and/or have used several production systems in the past and are therefore well versed in multiple user environments, can provide input and identify shortcomings in a prototype that, in general, may be difficult to identify during the benchmarking described in Section 4.5.

There are several levels of feedback that can be expected when engaging such experienced computational scientists in the prototype evaluation:

- **Realistic benchmark cases**
  Users are aware of the current state-of-the-art in their domains and, by extension, have certain expectations on the performance of their application on modern hardware. For example, a user will know what problem sizes or problem resolution is expected to be state-of-the-art within the next three to five years in their domain; the period during which a prototype is expected to be production-level technology. Advanced users will therefore benchmark a prototype with application software and with cases that can be more up-to-date than prepared kernel benchmarks.
- **Unconventional toolchain use**
  Full-fledged applications usually involve several software dependencies; in some cases involving different programming languages and SDKs. Advanced users that have experience compiling and deploying on production HPC systems will have certain expectations on the interoperability between different toolchains that may be useful to assess during a prototype evaluation. For example, some use cases may require a specific or custom version of an MPI implementation or may require combining a given MPI implementation with a specific compiler version.
- **Complex workflows**
  Realistic workflows used by users on production systems may be difficult to simulate during standard benchmarking. Modern complex simulations and analyses will include auto-generated scripts, check-pointing, multiple dependencies, and various pre- and post-processing steps that may not run on the same number of processes. Such use cases may reveal shortcomings in the overall prototype that may not be identified during benchmarking. Examples may include workflows which generate a large number of temporary files or with specific post-processing patterns which reveal I/O bottlenecks.
- **User environment**
  Users may have certain expectations of system-level software that can be difficult to identify outside a realistic use case but which may be non-standard on a prototype system. Examples include expectations on which environment variables are set and how the module’s environment is set up. If the purpose of the prototype evaluation is to assess the production level readiness of the system such feedback from users may be beneficial.

5 Conclusion

Prototyping projects in HPC are essential to evaluate new technologies, keep up-to-date with the latest developments in hardware and software, and assess their applicability and usefulness for production use. However, such projects are tricky to plan and execute and often the evaluation of the project and its outcome are neglected. As experience is the key to successfully run such a project, this best practice guide synthesized the experience gained from previous prototyping projects within PRACE and other FP7 and H2020 funding schemes. It is important to stress that each individual phase of such a project, as outlined in Chapters 2 to 4, is important for the success of the project. While this might be obvious for the planning and execution phases it is also the case for the evaluation: lessons learnt from previous projects will help to make the next even more successful.

Within the PRACE-1IP and PRACE-2IP projects, funding was available for small to medium sized prototyping efforts. The scope of the prototype systems procured within those projects ranged from small single node prototypes to large compute clusters with a budget of ~1M€. They were particularly useful for information gathering and sharing among the PRACE partners. Without funding from PRACE some of them would not have been possible and others would have been performed on a smaller scale and without involvement of other PRACE partners. With the move towards larger FETHPC co-design projects in FP7 and H2020, the funding for such small-scale hardware prototyping activities is no longer available. There is now a noticeable gap for these kinds of activities which could be complementary to the FETHPC co-design projects and be beneficial to the HPC community as a whole.

There will be an update to this best practice guide within PRACE-5IP; it will include tools to evaluate prototyping projects like potential KPIs, benchmarks, and a checklist for system and development tools.