# PRACE-6IP WP8 Project Proposal

| | |
|---|---|
| Project Title | QuantEx: Efficient Quantum Circuit Simulation on Exascale Systems |
| Project Leader | Niall Moran, ICHEC (Irish Centre for High-End Computing) |
| Project Contributor | Lee James O'Riordan, ICHEC (Irish Centre for High-End Computing) |
| Project Contributor | Myles Doyle, ICHEC (Irish Centre for High-End Computing) |
| Project Contributor | Venkatesh Kannan, ICHEC (Irish Centre for High-End Computing) |
| Project Contributor | Luigi Iapichino, LRZ (Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities) |

## 1. Project Abstract

In this project we will develop a platform consisting of modular quantum circuit simulation tools which use tensor network contractions methods, are capable of running efficiently on heterogeneous compute platforms and scaling to exploit pre-exascale and exascale compute resources. Modern development practices and software design methodologies will be used along with hierarchical layers of abstraction to encapsulate complexity and enable these tools to be easily extended and integrated into users' circuit simulation codes.

Classical simulation of quantum circuits is essential for debugging and validating the accuracy of quantum computing devices and algorithms. This is a very computationally demanding problem owing to the exponential growth in the state space as the number of qubits is increased. Up until recently it has been possible to simulate the largest prototype universal quantum computers using direct evolution of the quantum state (where the full wave-function is stored in memory (or on disk)) using modest personal computing resources. With the recent emergence of Noisy Intermediate Scale Quantum (NISQ) devices, it has become intractable to use this approach to simulate devices of this size on even the largest supercomputers.

The most promising alternative to direct evolution of the full wave-function are tensor network contraction methods. These methods were developed by condensed matter physics and quantum information communities to simulate systems of strongly interacting particles and offer greater efficiency and flexibility over standard approaches. They work by representing quantum states and operators using networks of tensors where calculations correspond to contractions over network edges. Recently, these methods were used in work on quantum supremacy experiments by NASA, ORNL and Google (http://arxiv.org/abs/1905.00444) to simulate particular types of random quantum circuits which would not have been possible using direct evolution of the wave-function. They achieved sustained performance of almost 300 (single precision) petaFLOPS on the Summit supercomputer at Oak Ridge National Laboratory. Despite this and similar efforts by Alibaba (https://arxiv.org/abs/1907.11217), there is a lack of open high level software which enables the simulation of general quantum circuits using tensor contraction methods on large heterogeneous clusters.

Due to the compute density and power efficiency of GPUs and the emergence of new processor architectures from projects like the European Processor Initiative, there is a trend towards more heterogeneous systems. This complicates the development of applications and necessitates the use of appropriate abstractions and interfaces to manage and encapsulate complexity and enable extensibility and sustainability.

## 2. Requested and committed resources

| Partner name | Central PRACE funding (PM) | Partner committed PMs | In-kind PMs |
|---|---|---|---|
| ICHEC (Irish Centre for High-End Computing) | 22 | 16 | 6 |
| LRZ (Leibniz Supercomputing Centre) | 11 | - | 11 |
| TOTAL | 33 | 16 | 17 |

## 3. The Team

### Description of the team

The team is composed of four individuals from ICHEC and one from LRZ. As evidenced in the short CVs below, the team has a range of complementary and relevant experience as well as a proven track record in similar software development projects for large-scale HPC systems. The project will be coordinated via 1. project group meetings and 2. meetings with stakeholders and end users.

1. Project group meetings involving all team members will be held monthly and will involve discussion of project activities, project progress and task prioritisation. For issues that are identified which require further discussion, additional meetings will be organised between the relevant team members
2. Meetings with stakeholders and end users will be scheduled to take place every six months. These meetings will serve to disseminate project progress, coordinate user engagement and afford end users the opportunity to give their feedback and direction

Section 5 provides a breakdown of the individual tasks, milestones and deliverables while section 7 describes the technologies and mechanisms that will be adopted to enable close collaboration, coordination as well as software sustainability and maintainability.

**Lead Applicant** - Niall Moran (www.ichec.ie/staff/niall-moran-phd)

Niall is a senior computational scientist at ICHEC (Irish Centre for High-End Computing). He currently works on developing and optimising computationally intensive applications on multi-core, many-node, GPU accelerated compute clusters. During his PhD and subsequent postdoctoral appointments he worked on numerical methods for treating many body quantum systems. He developed the DoQO and Hammer packages for simulating quantum spin and Fractional Quantum Hall (FQH) systems. These packages successfully scaled to thousands of cores on the Blue Gene/P JUGENE and SGI Fionn clusters and possessed novel features for exploiting quantum symmetries and entanglement structure. He has also made significant contributions to the DiagHam package, a large community code for treating FQH systems. In addition to this, Niall has worked extensively in industry developing software and managing teams of software developers and data scientists. This covered a broad range of domains from the development of automotive computer vision algorithms on custom embedded platforms to developing products which apply machine learning methods to high throughput network packet data for financial institutions and stock exchanges.

**Contributor 1** - Lee James O'Riordan (www.ichec.ie/staff/lee-oriordan-phd)

Lee is a Research Computational Scientist at ICHEC (Irish Centre for High-End Computing). Currently, he is investigating the application of quantum computation for use in natural language processing problems. He has worked on developing software solutions under the Exascale Computing Project (ECP) for applications in structural biology (ExaFEL) targeting NERSC's Cori supercomputer. During his PhD he has worked on investigating the dynamical behaviour of Bose-Einstein condensates, and is the lead author of the

GPU-enabled linear and non-linear Schrodinger equation solver suite "GPUE". He has experience developing for GPU, multi-core, and multi-node systems. Prior to this, Lee was a software developer at IBM, Dublin Software Labs.

**Contributor 2** - Myles Doyle (www.ichec.ie/staff/myles-doyle)

Myles is a Research Computational Scientist at ICHEC (Irish Centre for High End Computing). He is currently investigating the application of existing quantum algorithms in quantum computing, specifically for the use case of natural language processing. He completed his M.Sc. in High Performance Computing (HPC) after finishing his B.A. Mod. in Mathematics, both at Trinity College Dublin. Previous work involved developing and delivering course material for an Introduction to Deep Learning course, and research into dynamic tuning of HPC applications for energy optimisation.

**Contributor 3** - Venkatesh Kannan (www.ichec.ie/venkateshkannan)

Venkatesh leads the Novel Technologies Activity at ICHEC (Irish Centre for High-End Computing). His work involves investigating the use of conventional and emerging high-performance computing platforms and techniques for application to industrial, research and academic use-cases and projects. He has experience spanning programming multi-core and many-core HPC systems, energy-efficiency for exascale computing, programming quantum computing platforms, development of machine learning techniques and solutions for earth observation, transport and other such sectors. Venkatesh also coordinates the Quantum Programming Initiative at ICHEC, which is co-funded by the Irish state and industry partners, and targets building programming expertise and use-cases on currently available quantum computing platforms. Prior to HPC, Venkatesh has experience in modelling systems for simulation and analysis using Petri Nets, and formal transformation of functional programs for automatic parallelisation.

**Contributor 4** - Luigi Iapichino (iapichino.userweb.mwn.de)

Luigi Iapichino holds a position of application specialist for astrophysics and quantum computing at LRZ. He is the team lead of the LRZ Application Lab for Astro and Plasma Physics (AstroLab), and LRZ leader in quantum computing. In the PRACE framework, he coordinates the HPC high-level support at LRZ. Among his interests are code modernization for many-core and multi-core architectures, and quantum computing simulations on high-end HPC systems. He completed in 2005 his PhD in physics at the Technical University of Munich, working at the Max Planck Institute for Astrophysics. Before moving to LRZ in 2014, he worked at the Universities of Würzburg and Heidelberg, involved in research projects related to computational astrophysics.

## 4. Project description

It is envisaged that Quantum computing devices once mature will become core components of the HPC ecosystem. This will initially be as accelerator Quantum Processing Units (QPU) used in hybrid quantum-classical algorithms to enable significant speedup, make previously intractable problems tractable and/or increase power efficiency. The compute power of quantum devices derives from the quantum state space, the dimension of which grows exponentially with the number of qubits. This exponential growth makes simulation very difficult and underlies the need for flexible, efficient and scalable simulation methods.

Tensor network contraction methods are the most powerful methods known for simulating quantum circuits [1], [2]. From a HPC perspective, they have many advantages over direct evolution of the full wave-function. These include - much lower memory storage requirements, dense matrix matrix multiplication in place of sparse matrix operations and flexible slicing and decomposition possibilities which avoids the need for frequent global synchronisation and reduces dependence on fast internode connectivity. These methods have been demonstrated to scale on the largest available systems and are the subject of a growing body of literature and software.

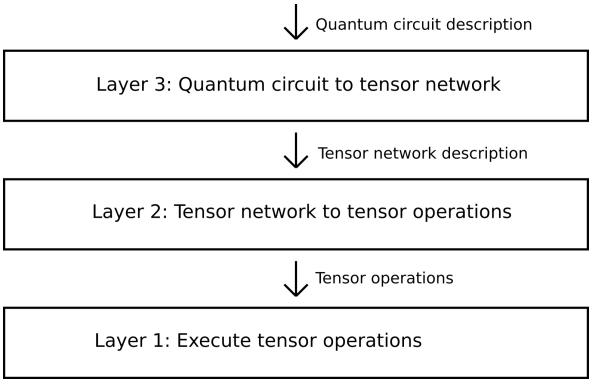
The objectives of this project are to:

1. Allow users to scale general quantum circuit simulations efficiently to exa-scale

2. Enable researchers and developers to easily integrate the resulting software in their own projects
3. Provide support for emerging processor architectures used in European pre-exascale and exascale machines

## Methodology

To meet the project objectives, we propose a framework consisting of three layers with well defined interfaces between them. Each layer will be made up of tools and components specialised for performing the tasks of that layer. Existing tools and libraries will be used and adapted where possible and well defined formats used for interchange between layers where appropriate. A hierarchical task based scheduler will be used to schedule tasks on groups of nodes, devices and cores and manage the transfer of data and results [3].




Here we describe at a high level how each of the layers operate and fit together.

### Layer 1

This layer provides low level functionality for performing operations on tensors. The core operations that this layer must support are:

- Movement of tensor data between different nodes and memory spaces within a node. Before applying any tensor operations, it is necessary to transfer the relevant tensor data to the local memory space. The overhead of this can be amortized by overlapping with computation and using preallocated memory
- Tensor contraction operations contract links between tensors. By transposing indices these can be recast as matrix matrix multiplication operations and standard dgemm kernels can be applied (MKL, ESSL, cuBLAS). It is nontrivial to efficiently transpose these indices, but algorithms and packages have been developed for this purpose (e.g. cuTT [4])
- Tensor compression operations which apply decomposition methods (QR or SVD) to reduce the dimension of links between tensors. These can only be applied for certain network topologies
- Decomposition of tensor contractions into collections of smaller tensor contractions using tensor slicing [5]. This can allow contractions to be performed where the memory requirements exceed that of a single device/node

The interface for each of these operations will in turn be well defined to allow specific implementations to be used on different hardware platforms. The TAL-SH library is capable of efficiently performing tensor operations on a single node using multiple CPUs/GPUs and of managing data movement between host and device as well as inter-device device transfers using RDMA. This will be used as the basis for this layer. Extensions will be made to perform calculations over multiple nodes using MPI-3 and support will be added for additional hardware platforms (FPGA, RISC 5) as required.

## Layer 2

This layer is responsible for performing high level operations on networks of tensors. This is achieved by decomposing these operations into operations on smaller numbers of tensors. These are then mapped to tasks performing layer 1 operations which are scheduled to run on the available resources. The operations of this layer include:

- Deciding the order in which to perform tensor contractions. Finding the optimal order is equivalent to finding the minimum-size tree decomposition of a graph and is believed to be NP-complete[6]. While finding the optimal order may be intractable in many cases, there are algorithms to find good contraction orders which will be integrated. The QuickBB[7] algorithm and software package can be used to find an approximately optimal decomposition in a finite amount of time
- Implementing noise models. It is possible to simulate the effect of certain types of circuit noise by performing tensor contractions using only samples of the indices. Considering noise in this way can actually increase the efficiency of the simulations and allow larger systems or more amplitudes to be calculated. It is possible to model other types of noise can be modelled by randomly inserting additional single-qubit gates

## Layer 3

This layer is responsible for translating the initial quantum circuit to a tensor network. There are many competing standards for describing quantum circuits (e.g. QASM[8], QUIL[9], XACC[10]). As part of the design of this layer, we will pick one of these which best suits our needs. Other formats can be supported by adding functionality to translate to the selected format. There are many trade-offs in choosing a qubit mapping and topology which depends on the particulars of each circuit, the aims of the simulation and target hardware. Initially this layer will use a small range of standard topologies (linear, 2D grid etc...) and allow the user to manually specify their chosen topology. If time allows we will look at integrating algorithms which can evaluate and automatically select the most appropriate topology.

## Existing software

There are many high quality software packages available which we intend to evaluate, adapt and integrate where appropriate. These include:

- **TAL-SH:** Tensor Algebra Library for Shared Memory Computers is designed to be used on single nodes equipped with multicore CPU, Nvidia GPU, and Intel Xeon Phi. This highly optimised library implements basic tensor algebra operations with interfaces to C, C++11, and Fortran 90+. GPLv3 license
- **qTorch:** Quantum Tensor Contraction Handler[6] provides functionality to read from QASM format, select an appropriate contraction order using QuickBB or a stochastic alternative and perform the contractions on multicore systems. Apache v2 license
- **TNQVM:** The Tensor Network Quantum Virtual Machine[11] plugs into the XACC accelerator and enables circuit simulation. On the backend it uses the ITensor C++ and Julia library. BSD 3-clause license
- **cuTT:** CUDA Tensor Transpose[4] is a high performance library for performing tensor transpose library for NVIDIA GPUs. MIT license.
- **ExaTensor:** is a basic numerical tensor algebra library for distributed HPC systems equipped with multicore CPU and NVIDIA GPU. It features a hierarchical task-based parallel runtime based on the virtual tensor algebra processor architecture, i.e. a software processor specialized to numerical tensor algebra workloads on heterogeneous HPC systems (multicore/KNL + NVIDIA GPU). Not yet reached production quality. LGPLv3 license
- **ExaTN:** Exascale Tensor Networks is a software library for expressing and processing hierarchical tensor networks on homo- and heterogeneous HPC platforms of vastly different scale, from laptops to leadership HPC systems. BSD 3-clause license
- **opt_einsum:** Python based package for performing contractions with optimised ordering[12]
- **quimb:** A python package for quantum information and many-body calculations using tensor networks[13]

- **TensorNetwork:** TensorNetwork is an open source library created by Google for implementing tensor network algorithms in TensorFlow[14]

## Technologies

The heterogeneous nature of the target architectures and pre-existing software necessitates a flexible approach to language selection. We will adopt the approach of using high level languages where possible to accelerate development and prototyping. Julialang[15] will be used as a glue language for its ease of integration with Fortran, C/C++ and Python components. We will also explore the possibility of using Julia for lower level performance critical component if it suits. The recent Celeste[16] and CLiMA[17] projects have demonstrated its effectiveness in large complex projects at PetaScale using NVIDIA GPUs. To ensure performance and scalability MPI-3, RDMA, CUDA, OpenMP and OpenACC will be used where appropriate. Interfaces will be designed such that the amount of device and technology specific code is well separated and kept to a minimum.

Our development activities will be strongly coupled with performance profiling, optimisation and testing on a wide range of modern HPC architectures. In the framework of PRACE applications for computing time we plan to apply for computing time at GCS/LRZ in Germany. This PRACE site hosts SuperMUC-NG, the largest HPC system in the European Union, consisting of more than 6000 Intel Xeon Scalable Processor nodes one of the currently outstanding computing architectures.

The performance profiling and optimisation will be focused on fully exposing parallelism at all available levels (including vectorisation), and in accessing data into memory in a cache-friendly and efficient way. Special care will be taken in keeping the adopted optimisation solution portable to different hardware flavours, whenever possible. The project partners at LRZ have a long-standing experience on code profiling and modernisation using the tools from Intel Parallel Studio [18], [19] and will coordinate the project sub-tasks related to these activities.

During the project we will also apply for computing access and code profiling on other platforms, like for example GCS/HLRS in Germany which in the near future will host Hawk, a multi-PetaFLOPS system consisting of next-generation AMD EPYC processors. This will nicely complement the analysis done on SuperMUC-NG and demonstrate to which extent our tools are capable to be ported on modern or upcoming architectures.

## Use Cases

Use cases will play a central role in this project. They will be used to inform the design of the framework, for testing, benchmarking and validation as well as serving as examples for users wishing to integrate the developed tools into their workflows. A range of use cases will be used to exercise different components of the systems. Among the use cases we plan to use are:

- **Random Quantum Circuits (RQCs):** These circuits are very challenging circuits to simulate and are thus useful for benchmarking purposes. Particular classes of RQCs were used in recent quantum supremacy experiments. They can be used for certified random generation, a potential application of NISQ devices
- **Variational Quantum Eigenvalue (VQE) circuits:** VQE algorithms can be used for quantum chemistry applications. These are the leading candidates for the first practical application of quantum computing and are thus very interesting circuits to simulate [20]
- **qNLP circuits:** The DisCo formalism [21][22] offers a method to infer relative meanings between sentences through encoding and processing in a quantum system. The required operations for these methods are mapped to the quantum circuit model simulated through use of the Intel QS[23] library. Running the qNLP circuit models on the QuantEx tensor network simulator will allow an ideal use-case comparison with the exact results offered by the Intel QS state-vector simulation. The comparison of both methods for circuit simulation will ideally result in publishable results
- **Model evolution of a quantum state in a "surface code" error-correction cycle:** Model evolution of a quantum state in a "surface code" error-correction cycle. Examining the effect of physical errors

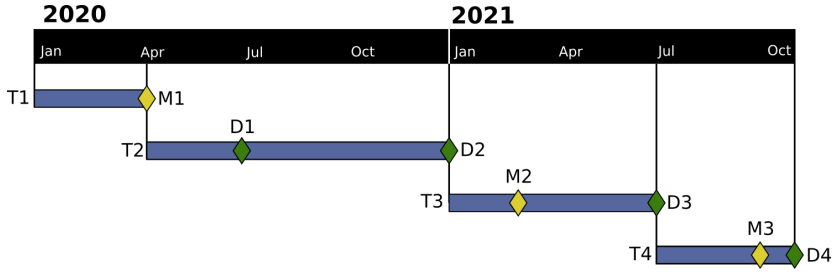
(which can be modelled on individual qubits) at the logical level will greatly aid with the development of “surface-code” error-correction methods [24]

### Tasks

The project will proceed in an agile manner with iterative delivery of features and frequent reprioritisation of tasks. Section 5 shows the expected timeline for tasks, milestones and deliverables. Here we provide additional detail on breakdown of tasks.

| Task | Months | Sub-tasks | Outputs |
|---|---|---|---|
| T1 | M01-M03 | T1.1 Evaluate existing software in terms of functionality and how it fits into proposed framework | M1 |
| | | T1.2 Evaluate existing software from performance and technology perspective with consideration given ease of integration with other components and potential performance bottlenecks | |
| | | T.1.3 Based on evaluations, design first implementation of framework and initial use cases to target | |
| | | T1.4 Prepare tests and use cases consisting of circuits covering a range of problem and circuit types. Prepare ground truth data where feasible (smaller circuits) using existing full wave-function simulators | |
| T2 | M04-M12 | T.2.1 Setup development environment with code repository, continuous integration and issue tracking systems | D1, D2 |
| | | T2.2 Prepare interim report with details of design decisions and project progress | |
| | | T2.3 Build initial implementation based on design arising from task and validate with tests and use cases | |
| | | T2.4 Package developed software, document installation and usage instructions and make publicly available | |
| | | T2.5 Prepare progress report detailing progress | |
| T3 | M13-M18 | T3.1 Gather feedback on prototype software release from end users. Identify bottlenecks, bugs, integration and usability issues as well as additional use cases | M2, D3 |
| | | T3.2 Perform details high level and per-component profiling to identify inefficiencies, scaling characteristics and opportunities for performance gains | |
| | | T3.3 Implement changes and make necessary updates to address issues raised and identified inefficiencies | |
| | | T3.4 Cleanup interfaces and add comprehensive documentation | |
| T4 | M19-M22 | T4.1 Perform final benchmarking runs on available architectures and largest accessible systems | M3, D4 |
| | | T4.2 Prepare final report with achievements, lessons learned and performance and scalability assessment | |

## 5. Project structure




| Month | Description of Milestones and Deliverables |
|---|---|
| M03 | M1: Finalised initial design of modules and interfaces, initial use cases defined |
| M06 | D1: Interim progress report: staff and project structure in place, prototype design decided and development successfully initiated |
| M12 | D2: Interim progress report & public prototype release: software ready for end user testing and evaluation, development infrastructure (repository, continuous integration etc...) in place |
| M14 | M2: User and stakeholder feedback collated, bottlenecks, improvements and final use cases identified |
| M18 | D3: Interim progress report & public production quality software release: documentation, testing results and issue tracker in place |
| M21 | M3: Benchmarking runs completed and results analysed |
| M22 | D4: FInal white paper report with achievements, lessons learned and assessment of performance on largest available systems |

## 6. Expected benefits now and impact in 5 years

The work carried out during this project is expected to have significant short and long term benefits. By enabling researchers working on quantum algorithms, the design of quantum hardware, error correction codes, circuit simulators and quantum simulation methods to make use of exascale resources, it will accelerate their research and potentially lead to high impact publishable scientific results and new discoveries.

The modular layer based design employed in this project will enable individual layers and components to be integrated into users' applications in this and related domains, e.g. the simulation of quantum systems in condensed matter systems. By building applications on top of the modules developed in this project, users will improve the separation of concerns and enable performance portability and scalability of their applications across a range of hardware architectures and system scales.

## 7. Software distribution

All software tools and use cases developed during this project will be released open source under a BSD style license. Software will be delivered to end-users through a variety of means.

- The code will be available on an open repository located on a GitLab instance operated by LRZ (https://gitlab.lrz.de). This service is operated as an official LRZ service, in order to ensure long-term access to software projects that are publicly funded. The repository will be mirrored on GitHub for additional resilience and to further ensure its long term availability
- To improve ease of use and integration by end users, developed software will also be distributed in packaged form along with dependencies (where licenses allow). For Python packages these packages will be installable with the python package installer pip or Conda the package management and

environment manager. For Julia packages, these will be installable via the Pkg.jl package manager. Docker and/or Singularity images will also be created and made available on Docker Hub or Singularity Hub as appropriate

- The use of issue tracking and merge request features of GitLab will allow users to submit bug, feature and enhancement requests as well as submit patches. This will facilitate the long term maintenance of the software beyond the end of the project

## 8. Software sustainability

Throughout the project, modern development tools and techniques will be used which are supported by industries beyond HPC. As indicated in section 7 above, a GitLab instance hosted at LRZ (https://gitlab.lrz.de) will be used to host the code during development and after release. This platform integrates a number of essential tools for software development and deployment. Among other functionalities, the LRZ GitLab allows issue tracking, which will be used for collaborative work management and bug tracking. In addition the LRZ GitLab has support for managing continuous integration and automated build and testing. The continuous integration pipeline will automatically build and test the software using a range of compilers and environments to validate and verify correctness. Sufficient unit test coverage, the integration of automated linting and static analysis will help maintain code quality, prevent hidden bugs and result in more standardised robust software. Automatic code generation tools like Sphinx and Doxygen will be used to ensure that documentation remains up to date.

## 9. Connections to users

To ensure that the resulting software and tools fit the changing needs of end users, we will remain in contact with the end users throughout the project. As mentioned in section 3, meetings will be held with all stakeholders and end users every six months. To facilitate closer alignment with the needs of users, we will work with them to come up with use cases in line with their needs and gather feedback on software releases. As described in sections 7 and 8, the use of a public repository as well as issue tracking and merge request features will allow users to engage more closely with the development by submitting issues, feature requests and patches.

As mentioned in section 4, the approach we will follow in this project is to integrate and adapt existing codes and libraries where possible. One particular project which has some overlap with the proposed project is the ExaTensor project at the Accelerated Data Analytics and Computing Institute (ADAC) (https://iadac.github.io/projects/). This project aims at creating user level API functions for running tensor algebra workloads on large heterogeneous compute clusters. We plan to work with the developers of this project to leverage the libraries and tools developed by this project, particularly the TAL-SH library.

## 10. Risk assessment (including personnel) and mitigation strategies

In the following table, risks for the project that are considered relevant for the proposed project are listed along with proposed control measures to mitigate them.

| Hazard | Risk | Risk Control Measures / Mitigations |
|---|---|---|
| Personnel | Key personnel leaving the project and taking critical information with them. | Increased collaboration and knowledge sharing within the team. Proper internal documentation and use of agile software development methods. |
| Tight timelines / | Project plan and intended timelines are planned tight. Unforeseen project | Use of incremental development and agile software development methods. |

| | | |
|---|---|---|
| Unexpected delays | delays can happen. | |
| Immature adopted technology | Chosen technologies or their implementation may be immature, unreliable or not fit for purpose. | Build up/make use of close connections to the companies/institutions, which are driving the development.<br><br>Apply prototyping / testbeds. Project specification which allow to achieve partial solutions / goals with established technologies. |