Challenge - DS07 Société de l'information et de la communication - PRCE

Solutions logicielles pour l'optimisation des programmes et ressources quantiques

Software-based quantum program and resource optimisation

Accounting summary:
→ 48-month (4 years) project
→ 554 k€ requested
→ 337 m.m (including 61 m.m from external fundings)

Table of contents

Project summary

Quantum computers can theoretically solve problems out of reach of classical computers. We aim at easing the crucial back and forth interactions between the theoretical approach to quantum computing and the technological efforts made to implement the quantum computer. Our software-based quantum program and resource optimisation (SoftQPRO) project consists in developing high level techniques based on static analysis, certification, transformations of quantum graphical languages, and optimisation techniques to obtain a compilation suite for quantum programming languages. We will target various computational model back-ends (e.g. QRAM, measurement-based quantum computations) as well as classical simulation. Classical simulation is central in the development of the quantum computer, on both ends: as a way to test quantum programs but also as a way to test quantum computer prototypes. For this reason we aim at designing sophisticated simulation techniques on classical high-performance computers (HPC).

This ambitious and original project lies at the interface of three main topics: quantum computation, programming languages and formal methods, and high-performance computing. The proposed consortium answers this challenge by bringing in together 3 complementary academic (LORIA, LRI, and CEA) and 1 industrial (Atos-Bull) partners, experts from diverse horizons, specialists in various aspects of these complementary topics.

The goal is to bridge the theoretical approaches of quantum computing and technological efforts, by developing a full, and certified, compilation chain in order to program the quantum computer. With this project we also aim at the diffusion of its outcomes by offering, among other actions, online IDEs to the

tools and methods developed along the project. As the simulation capability is key for the adoption by end-users, Bull commits to provide end-users a cloud-based access to the HPC simulation engine after the end of the project, subject to acceptance of terms of use.

Summary table of persons involved in the project:

Partner	Name	First name	Current position	Involvement (p.m)	Role & responsibilities (4 lines max)
LORIA	Perdrix	Simon	CR1 CNRS	24	Scientific coordinator
	Péchoux	Romain	MCF Univ. Lorraine	12	Other partner
	Marion	Jean-Yves	PU IUF, Univ. Lorraine, Head of LORIA	9.6	Other partner
	Jeandel	Emmanuel	Prof Univ. Lorraine	19.2	Other partner
	Hainry	Emmanuel	MCF Univ. Lorraine	12	Other partner
	Vilmart	Renaud	PhD	8	Non-permanent partner
	[to hire]		Postdoc funded by the present ANR call	12	Non-permanent partner
	[to hire]		PhD LORIA/LRI, funded by the present ANR call*	18	Non-permanent partner
	[to hire]		Graduate students funded by the present ANR call	12	Non-permanent partner
LRI	Valiron	Benoît	MCF CentraleSupelec	31	Scientific and technical leader co-PI
	Keller	Chantal	MCF Paris-Sud	5	Other partner
	Balabonski	Thibaut	MCF Paris-Sud	5	Other partner
	Baboulin	Marc	PU Paris-Sud	12	Other partner
	[to hire]		Postdoc funded by the present ANR call	12	Non-permanent partner
	[to hire]		PhD LORIA/LRI, funded by the present ANR call*	18	Non-permanent partner
	[to hire]		Graduate students funded by the present ANR call	12	Non-permanent partner
Bull	Allouche	Cyril	R&D Director	4.8	Scientific and technical leader
	Marchand	Bertrand	Research Engineer	9.6	Other partner
	Martiel	Simon	Research Engineer	14.8	Other partner
	Quintin	Jean-Noël	Research Engineer	14.8	Other partner
	Nguyen	Minh-Thien	Research Engineer	12	Other partner
	Goubault de Brugière	Timothée	Bull-LRI PhD, funded by CIFRE	12	Non-permanent partner
	[To hire]		Bull-LORIA PhD, funded by CIFRE	12	Non-permanent partner
CEA	Perrelle	Valentin	Research Engineer	3.5	Scientific and technical leader
	Bardin	Sébastien	Research Engineer	3	Other partner
	Bobot	François	Research Engineer, funded by CEA	5.5	Other partner
	[to hire][1]		post-doc, funded by CEA	12	Non-permanent partner

(*) The funding for the shared LORIA/LRI PhD is requested by the LORIA site.

[1] 12-month post-doct to be hired and work in collaboration with LRI partner

Any change that have been made in the full proposal compared to the pre-proposal

Since the pre-proposal, the consortium has evolved by adding a group of three researchers, Sébastien Bardin, François Bobot and Valentin Perrelle, from the Software Safety & Security Lab of CEA. They bring strong expertise in (classical) specification and verification techniques (abstract interpretation, symbolic execution, weakest precondition) as well as on the implementation of industrial-strength software analyzers (Frama-C, BINSEC). This is complementary to the already existing expertise of the consortium and of great value for the success of the project. They are usual co-workers of some of the partners of the consortium (e.g. J.Y. Marion and S. Bardin have co-supervised a PhD student very recently in the context of the ANR Binsec). J.Y. Marion and S. Bardin have an expertise in developing compilation tool chains. For example, they developed a full compilation/code analysis tool-chain for x86-malware. Moreover, together with the partners located at LRI, the Software Safety & Security Lab have initiated a few months ago a working group on certification of quantum programs to prepare the present proposal.

As the size of the consortium increases, the principal investigator Simon Perdrix will be assisted by Benoit Valiron who will participate to the administrative burden and will serve as an (informal) co-PI. Benoit Valiron will also ensure a local and efficient every-day management of the part of the consortium located at plateau de Saclay.

Regarding the budget, to address a remark of a pre-proposal reviewer, the budget of the industrial partner has been decreased, and mainly re-allocated to the new partner.

The first work-package has been splitted in two work-packages, to separate between two distinct aspects: the design of the language, and the development of a set of formal methods specifically tailored for quantum programs and quantum computation.

We also added two other work-packages: WP5, concerned with the management aspects of the project, and WP6, concerned with the dissemination aspects.

I. Proposal's context, positioning and objective(s)

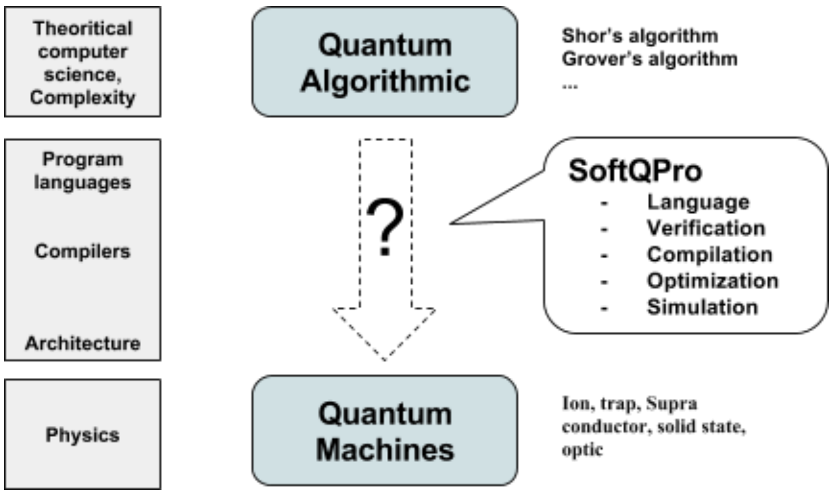
I.1. Objectives and scientific hypotheses

Context. If quantum computers can theoretically solve problems out of reach of classical computers, Quantum Computing is still in its infancy. On the hardware side, several technologies are competing (ion trap, supra conductor, solid state, optic…) for the implementation of a physical quantum computer. On the computational paradigm aspect, the situation is not clearer: if Knill's quantum random access machine (QRAM) model [K96] is the reference model for the design of quantum algorithms, several other promising models such as measurement-based quantum computation (MBQC) and adiabatic quantum computing are in the competition, and it is not yet clear whether one model would rule them all, or if each paradigm has its strengths and weaknesses making it more or less suitable as abstraction over a concrete, physical representation of qubits. Finally, and partly for these reasons, quantum computation is also immature on the concrete implementation of quantum algorithms. Indeed, if recent years have seen the advent of several scalable quantum programming languages (Quipper, LiquiD, …), they only offer partial solutions with respect to compilation, specification and verification of code and optimization schemes.

Nevertheless, from the perspective of concrete implementation of algorithms and analysis tools, quantum computation is in a better state nowadays than conventional computer was at its dawn: decades of research in the development of formal methods have yielded powerful techniques and methodologies to address the compilation of optimized and verified programs written in high-level languages, alleviating a bit the burden and the cost of debugging. With partners on both sides of the bridge, we plan to harvest these conventional techniques to design a novel and modular compilation toolchain for quantum computation.

Problem and challenges. In the context of quantum computation, there have been a few attempts in the developments of high-level methods in terms of programming languages [**VRS+15**,Liq,JPK+15], partial analysis tools [**P08**,DLZ10,YYW17], ad-hoc low-level representations of code [ProjectQ]. Some of these tools try to address the hardness of debugging programs by offering primitive emulation environments [Liq]. *However, these propositions are usually partial, isolated propositions*. A successful development of the quantum computer requires theoretical advances in quantum formal methods from which can be distilled a *quantum focused compilation toolchain*, together with an integrated study of available HPC tools to target quantum emulation.

Goal. We aim at *filling the gap between the theoretical quantum algorithms and the various underlying technologies* by *designing and providing a quantum focused compilation toolchain from dedicated high-level language to quantum emulation*. Our *software-based quantum program and resource optimisation (SoftQPro)* project aims at easing the crucial back and forth interactions between the theoretical approaches to quantum computing and the technological efforts made to implement the quantum computer. It follows a path used in the development of compilation techniques in conventional software: design of a language, with formal methods, static analysis tools and a set of optimizations, together with a backend for code execution. The language developed in the project will be called QuaML and the identified backend will be quantum emulation. We identify five crucial objectives:



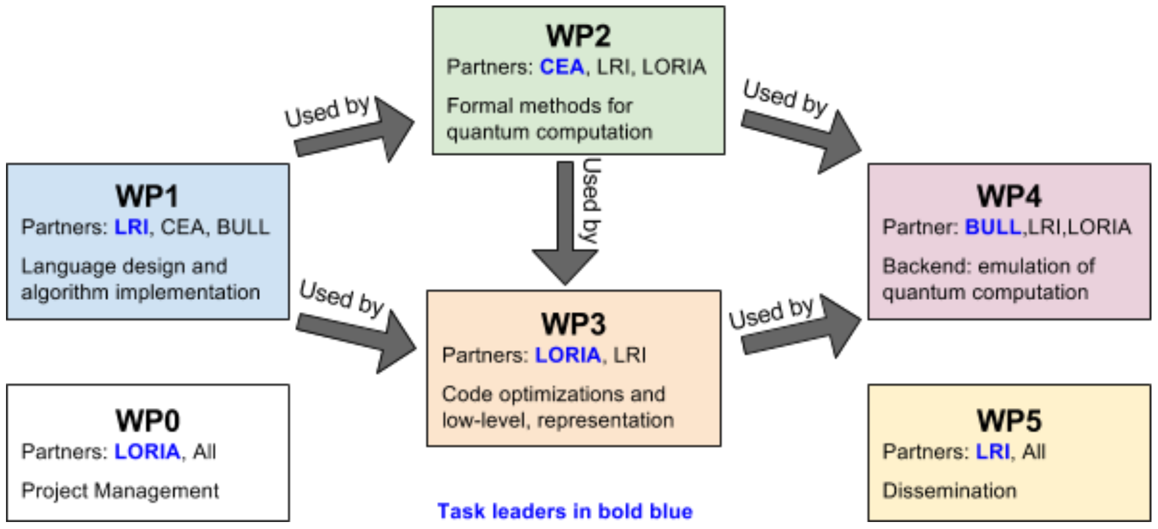
Objectives	**Measures of success**
G1. *A high level language (QuaML) for quantum algorithm design*	*Number of types of encodable algorithms*
G2. *Certification and resource analysis of quantum programs*	*Efficiency and accuracy (both theoretically & against a testbench to be decided upon)*
G3. *Optimization and equivalence of quantum codes*	*Efficiency and accuracy (both theoretically & against a testbench to be decided upon)*
G4. *Classical emulation-platform of quantum programs*	*Efficiency of the simulation / scalability*
G5. *Communication on the importance of quantum software*	*Popularity, citations*

These objectives form the basis of the work-packages described below and detailed in section II.3. Colors point out the correlations between objectives, workpackages, delivrables and final products.

Originality. The project is innovative in its form as it is one of the first ANR collaborative projects with an industrial partner in quantum computer science. It gathers partners specialists in conventional and in quantum methods having managing successful collaborations or having ongoing plans for collaborations, independently from the SoftQPro project. Regarding the state of the art, some papers [PRS15,**SRSV14**,YG07] address the question of the optimization of the circuits produced by compiling programs written in Quipper or other languages. We go beyond this with a compiling pipeline, two targeted paradigms of quantum computation (QRAM and MBQC) and an intermediate language. The project is also innovative on the methods: only a few proofs of concept on the use of static analysis for quantum programs have been published (entanglement analysis [**P08**], implicit computational complexity [DMZ10]), we aim at refining these techniques leading to efficient tools for program analysis.

Our approach. Any quantum algorithm can be expressed in a high level quantum programming language: a large library of quantum algorithms [**GLR+13**,**VRS+15**] have been written in Quipper, an open source language co-developed by Benoît Valiron (a partner of the project). In Quipper, only compilation to quantum circuits has been considered so far, but they can potentially be compiled for other targets depending on the computational model (unitary-based or measurement-based) and the hardware technology. Another interesting target is a simulation on a classical (high performance) computer. Our project aims at developing software-based quantum program and resource optimizations. The resources to optimize are typically the number of instructions, the size of the memory, the depth (parallel time), or the amount of entanglement for instance.

Our approach is based on a compilation pipeline of the QuaML language, formal language based on Quipper. The pipeline goes from QuaML programs to an intermediate language (ZX-calculus) and then to various paradigms of quantum computation with an emphasis on HPC simulation. It consists in six work-packages whose interdependence is represented by the following chart. The content of work-packages is briefly summarized in the next paragraphs (see Section II.3 for a complete description).


WP2
Partners: CEA, LRI, LORIA
Formal methods for quantum computation
WP1
Partners: LRI, CEA, BULL
Language design and algorithm implementation
WP4
Partner: BULL,LRI,LORIA
Backend: emulation of quantum computation
WP3
Partners: LORIA, LRI
Code optimizations and low-level, representation
WP0
Partners: LORIA, All
Project Management
WP5
Partners: LRI, All
Dissemination
Used by
Used by
Used by
Used by
Used by
Task leaders in bold blue

Work-package 1. **Design of the high level quantum programming language QuaML.** At the front end, we aim at improving the quantum programming environment Quipper by formalizing the language into QuaML, rationalizing and developing the existing libraries, and by developing input and output drivers for use in the compilation toolchain.This work-package is the entry door to the rest of the project, and is taken care of by the partner LRI and CEA.

Work-package 2. **Formal methods for quantum programs.** Design of modular certification tools for code assertions and contracts. High level verification techniques based on program static analysis like abstract interpretation or implicit computational complexity to optimize various resources. These techniques are successfully and massively used for classical code optimization and resource saving, whereas only a few proofs of concept exist in the quantum case [**P08**,DMZ10].

Work-package 3. **Intermediate graphical language.** Our novel approach relies on the use of an intermediate formal graphical language, the ZX-calculus [CD09]. Equipped with a powerful equational theory, it is an ideal common denominator for resource optimization. This work-package feeds from the results made in WP2, and is tackled by LORIA and LRI.

Work-package 4. **Back-end and emulation on classical super-computer.** For the back end, we plan to stay at the level of the computational paradigm and perform an analysis of quantum algorithms within the QRAM and the MBQC model of computation. Specific optimizations depending on the chosen computational model will be considered. Finally, a peculiar attention will be paid to emulation by means of HPC. In particular, this practical back-end will provide a complete compilation and execution platform together with specification, verification and testing tools. This last work-package is the hand of Bull and LRI (M. Baboulin).

Work-packages 0 and 5. **Management and dissemination.** WP0 is reserved for the interaction with ANR and the associated duties, whereas WP5 concentrates the communication and dissemination aspect of the project.

Final product and expected results. We expect SoftQPro to produce a solid and modular compilation toolchain from the ready-to-use high-level language QuaML down to a state-of-the-art, scalable emulation platform. As a whole, by harvesting from both conventional and quantum research areas, SoftQPro will bring both theoretical insights and practical advances to the field of quantum computation. We list the expected results for each described objective.

Objectives	**Final products**
G1	*A complete compilation toolchain from the high-level QuaML down to computational backends.*
G2	*A comprehensive solution for implementing and analyzing quantum algorithms.*
G3	*Paradigm dependent optimisation techniques ; theoretical results on equivalence of code*
G4	*An emulation toolbox with specific optimizations to make it as efficient as possible.*
G5	*A project website gathering all results, online IDEs and benchmarks with a thriving community of users*

I.2. Originality and relevance in relation to the state of the art

Background. The promises of quantum computation have sprung many hopes for solving problems of size unreachable from conventional computing. However, many challenges are still open and impair the concrete implementation of these promises. The problems range from the physical realization of quantum computer to the implementation of quantum algorithms: SoftQPro aims at providing a comprehensive solution for the latter side of the spectrum.

The issues on the software side of quantum computation can be summarized using the analogy of the compilation toolchain we aim at in SoftQPro. Let us define these problematics following this scheme and review the current state-of-the-art partial solutions, with contribution of partners & Preliminary results. The work-packages presented in the previous section are direct reflection to this presentation.

On programming languages and compilation toolchain. The first problem is the coding of quantum algorithms: it requires a programming language. After a decade of toy-languages [Gay,**SV06**,O09,AG09] aimed at testing various operational settings and presentation, nowadays the big players of the field are

- Quipper [**VRS+15**] : embedded in Haskell. Monadic operational semantics enforced by Haskell's type system. Valiron (LRI) is one of the developer of the language.
- LiquiD [Liq] : embedded language in F#. If it offers a compilation toolchain down to an emulation backend, the compilation is monolithic and does not allow easy modification or extension.
- QASM + Python [ProjectQ]: An ad-hoc low-level description language for logical circuits together with a Python library.
- ScaffCC [JPK+15]: C-like programming language compiling down to LLVM. Offers some simple optimizations, but the semantics of the language is very weak: un-parameterized static circuits.

On specification and verification. Once an algorithm is coded, the question is to make sure that the encoding effectively corresponds to the algorithm. Several preliminary results exist in quantum computation towards this goal. We also mention the relevant conventional techniques that are used in verification of conventional programs.

- Static analysis of resources [**MP09**], in particular entanglement [**P08**]
- Quantum Implicit Computational Complexity (ICC) [**GMR12**,DMZ10]
- Logical resource estimation of quantum programs [**SRSV14**,**SVM+17**]
- Quantum Hoare-style specification [HP06, YYW17].
- Denotational semantics [**PSV14**]: theoretical apparatus, not really scalable.
- Classical Verification: Why3 [**BFMP11**,**BFMP15**], Coq [Coq] and Isabelle/HOL [Isa],
- Classical Verification: symbolic Execution [**DBF+16a,DBF+16b**]
- Quantomatic [K12] : based on Isabelle [Isa], proof assistant for the ZX-calculus.
- Quantum model checker [GNP08]

On low-level representation and code optimization. In a conventional state-of-the-art compilation toolchain such as LLVM, the high-level language is translated down to an intermediate representation (IR), closer to a potential hardware. In the case of quantum computation, IR are currently either logical circuits (e.g. in Quipper), or some ad-hoc formalism (e.g. QASM). SoftQPro aims instead at the promising ZX-language [CD09,**DP09**,**BPW16**,DD16], developed from an categorical analysis [AB04] of the structures of Hilbert spaces required for quantum computation. This language provides an abstraction where programs are represented as graphs, and it naturally compiles down the chain back to circuit (and therefore to the QRAM model), but also to the alternative MBQC computational paradigm. Several problems are still open (partial approaches are cited):

- Expressivity of ZX calculus : towards a complete diagrammatic language [**JPV17**,**JPV+17**]
- Adding control flow to ZX-terms, maybe with !-graphs [KMS12], following the spirit behind dynamical circuits [**GLR+13**] and Geometry of Interaction [**LCVY17**].
- Optimization of reversible circuits [PRS15] are not scalable in general.

On paradigms of computation. The computational paradigm behind quantum computation is not completely settled yet. In particular, because of the wide differences in the quantum hardware capabilities, the circuit and QRAM [K96], and the MBQC models are still competing. For MBQC, the following questions and results are meaningful for SoftQPro:

- Characterisation of valid deterministic MBQC [**BKMP07**,**MMP+11**]
- MBQC parallel-time optimisation, provably more efficient than quantum circuits [**BKP10**]
- Rewriting strategies of ZX-terms [**DP10**] and relation with MBQC [CK17]
- There exists MBQC-based circuit optimization [PDK15] linking both paradigms. How do they scale for concrete examples?

On high performance simulation of quantum circuits. Along the development of a conventional program, debugging and testing can play a large role in correcting and validating a given implementation. In the case of quantum computation, both because of the current state of the hardware and its inherent limitation, this aspect can only be performed using emulation techniques on High-Performance Computing (HPC).

- The biggest current simulation is Intel's qHipster experiment from February 2016 [SSA16] with 2000 compute nodes.They report simulation of 40 qubits.
- TU Delft also have an emulation platform: QX simulator [QTech]
- Google (2015-2016) has a highly parallel simulation on GPU [Gpu]
- One important question is the investigation of hybrid systems HPC-quantum. [RLA+16].

Relevance / Position of the project. Quantum computation is currently thought of as a promising technology by many institutional, academic and industrial actors *worldwide*. The British NQIT and Dutch QuTech programs, the announced European flagship on quantum technologies, and the availability on the market of new (and more convincing) D-Wave machines are witnesses that quantum computing is entering a new area. At the national level, *Atos-Bull publicly announced in 2016 they are embarking on the quest of the quantum computer*. CEO Thierry Breton supported the european flagship to EC President Juncker and Atos-Bull is represented at the flagship High Level Steering Committee in the person of *Cyril Allouche*, Atos-Bull R&D director and partner of the present project. In July 2016, *Simon Perdrix* (coordinator of the present project) co-organized a workshop at ministry of research which emphasized, according to T. Calarco (Flagship leader) and P. Indelicato (*Dir. Cabinet* of Minister T. Mandon), the strength of the french community in the computer science approach to quantum technologies.

The focus on quantum technology has driven a strong interest from academics and industrials alike to use of quantum algorithms to solve practical problems yet unreachable using conventional technologies. One important barrier to lift lies on the software level, and consists in being able to actually encode the algorithms at sake. Going from a mathematical presentation of an algorithm to a concrete implementation is not a trivial task and in the classical world it makes use of sophisticated compilation toolchains. If in the realm of quantum computation the few existing ones are still incomplete and experimental, this subject is the focus of an increasing research effort. The first effort can be dated back to 2011 with the start of the IARPA-funded project QCS [Q2011] to which one the SoftQPro partner participated in (Valiron) [**GLR+12**]. Several languages for quantum computation sprouted from this project, among which Quipper [**VRS+15**] and ScaffCC [JPK+15]. In the years following the end of the project others local efforts began, yielding e.g. LiquiD at Microsoft and a python-based library [ProjectQ] at ETH and IBM.

The SoftQPro project is therefore positioned within a growing ecosystem of development. Its theoretical and practical aspects will feed the community and, capitalizing on the complementarity of the consortium, it will propose a novel compilation toolchain going beyond the current state of the art.

I.3. Risk management and methodology

SoftQPro is clearly a highly challenging project. Yet, we rely on a set of mitigations and fallbacks to help master these risks.

Methodology. In the approach followed in SoftQPro each one of WP1-4 can be first considered independently from the others, and builds up on existing technology that can be leveraged. *This means that an unexpected delay in one WP will not hold back another one*. For example, in case of delay in WP1, the formal methods of WP2 can first be considered with toy languages before moving to QuaML. The work on ZX and development of optimization methods in WP3, and the simulators developed in WP4 can be analyzed by themselves, and the circuits used to benchmark the techniques can be distilled from the existing language Quipper, and integrated to WP1 in a second step.

This *relative independence* among the work-packages was carefully design to reduce the risk of global failure: the worst that could happen is degraded performances and outputs on one particular aspect.

In order to even further restrain the risks of local failures within work-packages, in most cases *we make sure to propose alternative, complementary approaches*. For example, in the case of the development of optimization for ZX terms, we propose both to consider specific rewrite strategies and relation with circuits, to be able to capitalize on existing techniques and propose a solution nonetheless, even if degraded. The same happens in verification and analysis, where tradeoffs are a common lever.

Attainability of objectives. Each class of tools that we propose to use is already at least *partially developed and has demonstrated its suitability* for use-cases close to SoftQPro. In particular, Quipper already allows encoding of algorithms for nontrivial size problems, and comes with a library of all the main quantum algorithms already encoded in this language. Valiron, member of the partner LRI, participated in this effort. Also, static analysis, abstract interpretation, implicit computational complexity and code optimization techniques are used in the compilation of conventional languages. Worldwide specialists of classical static analysis [**GMR12**,**MP09**] are involved in this project (partner LRI, LORIA and CEA). The ZX-calculus is a well developed formal graphical language with a powerful equational theory which has been already successfully used in several domain of quantum information theory (quantum protocols [AB04], measurement-based quantum computation [**DP10**], foundations [CDKW12]). Finally, Quantomatic is a versatile, dedicated open source software allowing ZX-diagrams representation and transformations, where strategies can be defined in the software.

The consortium has been designed as a *small core of partners with backlogs of previous or ongoing* *collaborations*. The expertize of all partners altogether *spans the whole spectrum of the project*. The inclusion of CEA as partner within SoftQPro is an additional guarantee, as quantum computation is a strategic axis of this institution. We plan reinforced collaborations on each WP, witnessed by the already financed and the requested post-docs and Ph.D students.

Adequate resources. For SoftQPro we carefully planned the resources for the project. We ask for 4 years in order to give us the time to find suitable Ph.D and post-doc candidate to work with LORIA and LRI. We also ask funds for the organization of regular meetings amongst the partners, again to foster internal collaboration.

II. Project organisation and means implemented

II.1.Scientific coordinator

Simon Perdrix (PI) is a 36 year old CNRS researcher. He strongly contributed to the development of the ZX-calculus: from the early age [**DP09**,**DP10**], when he was working in the quantum group in Oxford when and where the ZX-calculus has been invented [CD09]; to the recent developments of the language, answering one of the main open question in this field [**PW16**]. By introducing the first static analysis techniques for quantum programs at SAS'08 [**P08**], he made a proof of concept of the software-based approach aimed to be refined in the project. He is also an expert of Measurement-based Quantum Computation one of the most promising model for a physical implementation of the quantum computer. He has over 40 refereed publications (ICALP, QIP, MFCS, ISAAC, SAS, FCT...). He leads the Quantum Computation french network (GT IQ) at CNRS GdR IM and is board of the CNRS Quantum Technology network (GdR IQFA). He has been PI of several projects (PEPS, Region Lorraine), and led work-packages in ANR and EU STREP projects. In 2016, he has been elected scientific secretary of section 6 at CoNRS. Section 6 is in charge, among other expertise duties, of hiring, promoting, and evaluating CNRS researchers in computer science.

Benoît Valiron (co-PI) is a 36 year old assistant professor at CentraleSupelec. He is one of the pioneer in the semantics of higher-order functional quantum programming languages [**SV06**,**V08**] that he studied and developed during his Ph.D. He then had the chance to investigate several related systems such as algebraic lambda-calculi [**V13**,**ADP+14**] and Geometry of Interaction [DFVY17] During a 2-year post-doc in the U.S., he became one of the main developer of the first scalable quantum programming language, Quipper [Quip]. With over 20 refereed publications, he has gained expertise in semantics of quantum computation and in related models of linear logics. He also built an expertise in types systems and formal methods and tools: Coq, Isabelle/HOL.

The role of Simon Perdrix and Benoît Valiron in the project SoftQPro is the standard role for PIs and is summarized in WP0: keep track of attainments and progress, and monitor each work-package and partner involvement. In case of failure or other delay in a sub-part of the project, They will gather input and ideas and be leader to choose an alternative path for success. Finally their other main responsibility will be to manage the dissemination aspect of the project and organize the associated workshops (see WP5).

II.2.Consortium

The project SoftQPro lies at the interface of three main topics: quantum computation, programming languages and formal methods, and high-performance computing (HPC). The proposed consortium answers this heterogeneity by bringing in together 3 academic (LORIA, LRI, and CEA) and 1 industrial (Atos-Bull) partners experts from diverse horizons, specialists in various aspects of these complementary topics.

- LORIA is the computer science lab of CNRS, Inria and Universite de Lorraine. Bolstered by the 500 people working in the lab, its scientific work is conducted in 27 teams including 16 common teams with INRIA. LORIA is today one of the biggest research labs in Lorraine. It brings to SoftQPro experts in quantum computing, ZX-calculus, measurement-based quantum computing and program analysis, including experts in (classical) implicit computational complexity.

- LRI is the computer science lab of Universite Paris Sud, and member of Digiteo and Systematic. The research themes addressed by LRI cover a wide spectrum of computer science focused on software ranging from fundamental to applied research. SoftQPro brings the complementary expertise of three teams of the laboratory: quantum programming language, (Modhel), formal methods and compiler architecture (VALS), and HPC (Parsys).
- Within CEA LIST (Paris Saclay, Computer Science division of CEA), the *Laboratory for Security and Safety* (LSL) focuses on software verification and formal methods. CEA brings a strong and wide expertise in (classical) program verification (deductive verification, symbolic execution, abstract interpretation), as well as a unique expertise in building industrial-strength software analyzers (tool Frama-C, prototype BINSEC) and industry transfer (Airbus, startup TrustInSoft).
- Bull, a subsidiary of the Atos, is the sole HPC manufacturer in Europe, and has built several of the 50 most powerful supercomputers in the World. Beginning of 2016, it launched its R&D program in quantum computing, with objectives to prepare the next generation of HPC systems, embedding quantum power. Its R&D team is located in Les Clayes sous Bois (78), and represents 10 full time, among them 3 PhD students..

The involvement of each partner of the consortium distributes on the work-packages as shown in the following table (in man.months over the four years)

	WP1	WP2	WP3	WP4	WP5	WP0
BULL	6 m.m			69 m.m	4 m.m	1 m.m
CEA	9 m.m	13 m.m			1 m.m	1 m.m
LORIA		46 m.m	63.8 m.m	8 m.m	5 m.m	4 m.m
LRI	50 m.m	25 m.m	3 m.m	10 m.m	4 m.m	3 m.m

Budget. HPC platform will be made available to all the partners by Atos-Bull. A small additional material investment will however be needed. Meetings between the partners will be organised to help to develop synergy. To supplement the workforce at the two academic partners, a support for a co-supervised PhD on quantum program static analysis, and two postdocs positions are requested: one to reinforce compilation and HPC simulation at LRI, and one at LORIA on optimisation and transformation in the intermediate language.
One of the reason for asking for 4 years of project instead of, say, 3, is because of the hiring of a PhD: this will give us some latitude to find a suitable candidate and still have funding for the whole period of studies.

	Postdoc	PhD	Internships	Conferences	Visits Consortium	2 laptops	Workshop kickoff m.	Overhead (8%)	Total
LORIA	59.4k€	105.5k€	6k€	36k€	6k€	3k€	7k€	17.8k€	240.7k€
LRI	50.1k€		6k€	30k€	5k€	3k€	2k€	7.7k€	103.8k€

	Total	Comments
Bull	160k€	25% of 56 m.m+5k€ for travel+50k€ of consumable (FPGA Stratix cards, HPC blades) ie 25% of 640k€
CEA	50k€	40% of (6.5 m.m + 6.7k€ for travel+1.5k€ computer)

A total budget of 554.5k€ is requested over 48 months: 394.5k€ for academic partners, 160k€ for Atos-Bull. LORIA and LRI will have 3k€ for consumable (computers), 6k€ for graduate level interns, and the funding for 1-year postdoc. Regarding travel, the funding for the conference trips (resp. visits

within the consortium) is based on 1.5k€ (resp. 350€) per trip, 6 trips a year for LORIA and 5 trips a year for LRI. The coordinating site asks for 105.5k€ for a co-directed PhD (LORIA/LRI) and 7k€ for organising an international workshop. The kick-off meeting (2k€) will be organised by LRI. CEA asks for 50k€ in total, including 6.5 m.m permanent staff, travels (6.7k€) and one laptop (1.5k€). Atos Bull requests 160k€. The requested funding is recapped in the above tables.
Finally, it is worth emphasizing that one Ph.D (CIFRE Bull-LORIA) and one post-doc (CEA) are planned to work on the project without funding requested, as well as 5.5 m.m of permanent staff (CEA).

Complementarity of the partners. If this consortium has been created specifically for this project, there have already been several previous successful academic collaborations amongst members, as summarized in the following table:

	BULL	LORIA	LRI
CEA	▨	previous collaboration	previous collaboration planned post-doc
LRI	CIFRE PhD	previous collaboration	▨
LORIA	on-going collaboration	▨	previous collaboration

In particular, LORIA and LRI have already written joint publications [**ADP+14**,**DPTV14**] and participated in common project (Stic AmSud FoQCoSS). LORIA and CEA also have written joint publications [**BDM17**,**DBF+16a**,**DBF+16b**,**DBF+16b**] and participated in common project (ANR Binsec on low-level malware analysis). A Bull-LRI CIFRE PhD started this fall on quantum programs and HPC-based quantum circuit simulation. LORIA and Atos-Bull have an ongoing collaboration, a CIFRE PhD is planned to be hired in the next few months. CEA and LRI are having a fruitful collaboration on program verification (Frama-C, Why3, ANR SOPRANO, ...) and have recently started collaborations on quantum verification and are in the process of hiring a 12-month post-doc aimed at SoftQPro (no funding requested, paid by CEA).

II.3.Means of achieving the objectives

The project can be mirrored to a compiler construction: a front-end for easing implementation and reasoning on quantum algorithms, internals for analysis and resource optimization, and finally various backends: computational models and simulations. The four workare -packages we propose for SoftQPro reflect this organization: one work-package for the front-end, one for the backend, and two for the middle layers: formal analysis and optimization. The work-packages are described in details below.

WP0: Management

WP leader: LORIA (Perdrix) **WP collaborators:** All

WP0 coordinates the project, deals with consortium agreement issues, sets up a cooperative framework (git, wiki, mailing list), and reports to ANR. We plan to have periodical steering meetings (email, phone, visio); general physical meetings will occur every year. Technical meetings focused on particular issues will be organized on-demand during the project.

Deliverables.
WP0.1 - T0+3 - Kickoff meeting
WP0.2 - T0+6 - ANR starting report
WP0.3 - T0+12 - Annual meeting
WP0.4 - T0+24 - Annual meeting
WP0.6 - T0+48 - ANR final report

WP1: Front-end: High level quantum programming language design (QuaML).

WP leader: LRI (Valiron) **WP collaborators:** CEA, BULL

Before being considered as potentially practical, quantum algorithms were first thought of and designed as tools for complexity analysis. Recently, as mentioned in the SoA several scalable quantum programming languages have been designed to implement concrete instances of quantum algorithms. For the project SoftQPro, we find Quipper to be the best candidate language to build on. Its operational semantics is well-defined enough to be formalized, yet sufficiently expressive to characterize all possible quantum algorithms. It is mostly lacking in back-ends and in language processing. But it is easier to add such back-ends than on other candidate languages.

Task 1. From embedded language to the formally defined QuaML.

Right now, Quipper is a programming language embedded in Haskell. If Haskell is a good host language, it only enforces policies encoded within its type system. The current presentation of Quipper is thus unsuited for rigorous code processing, analysis and certification.
Objective. Define a new formal language QuaML based on Quipper to allow the use of formal methods.
Methodology. Two complementary approaches: an abstract lambda-calculus with a resource-sensitive type system and native, explicit terms for circuits following the approach for ProtoQuipper [R15], and as a language embedded in Why3 [**BFMP11**,**BFMP15**] for the higher level proofs as done by Krakatoa (Java), Jessie (for C), Spark (Ada) .
Measure of success. The ability to be able to encode in QuaML the existing corpus of Quipper programs within the designed abstract syntax and semantics.
Deliverable. D1.1 - T0+0 → T0+9 - Report. Formal description of the language QuaML

Task 2. External libraries: Interacting with the world.

Quipper being an embedded language makes it particularly easy to use Haskell's contributed libraries for all the needed classical processing: prime numbers management, matrix diagonalization… This is one of the strength of embedded language, and the general goal of this second task is to recover this aspect in the formal presentation of the language.
Objectives. Capture within QuaML the pre- and post-processing done in the existing implementations of quantum algorithms in Quipper.
Methodology. Set up a notion of *interface* for QuaML. The desired constants and operations are provided as black-boxes in the abstract syntax, and the formal semantics of these black-boxes is provided axiomatically.
Measure of success. Against current quantum programs implemented in Quipper.
Deliverable. D1.2 - T0+9 → T0+18 - Report. Interfaces for the formal language QuaML

Task 3. Native libraries: Comprehensive and formal-method friendly

Quipper presently has a large corpus of internal libraries for various aspect of quantum algorithms gathered in several places.
Objectives. Rationalize and centralize these heterogeneous libraries. Recode them in QuaML in a formal-method friendly way: their constituents need to be well-typed, and to have a clear and well-defined semantics.
Methodology. First proceed by inspection and do the job for Quipper. Once convincing enough, proceed by level of difficulty: first the circuit combinators, then the arithmetic libraries, finally the synthesis tools.
Measure of success. Against current quantum programs implemented in Quipper.
Risk management. The main risk lies in porting the synthesis tools: as this is marginally used in the other WPs and tasks, failure would only slightly degrade the result.
Deliverables. D1.3 - T0+9 → T0+24 - Report and open-source code. Incorporation of Quipper's libraries within QuaML

Task 4. Down the chain: Compilation of QuaML to Intermediate language

This task is concerned with the final part of the front-end of the compilation tool-chain.
Objective. Provide an explicit internal representation for QuaML programs using the ZX-calculus developed and used in WP3.
Methodology. Ideally the internal representation should be as compatible as possible with the one in

Quantomatic. We plan to follow this approach to design our internal representation. We first focus on static ZX-terms which can already represent quantum circuit, and move to dynamic terms when D3.1 is completed
Risk management. The success of these objectives is tightly linked to D3.1. In case these were to fail, a fallback solution consists in considering static ZX terms. The overall risk for this particular task is low: it is essentially a matter of coding it.
Deliverable. D1.4 - T0+9 → T0+24 - Open-source code of an implementation of QuaML.

WP2: Specification, Analysis and Verification

WP leader: CEA (Perrelle) **WP collaborators:** LORIA, CEA, LRI

This work-package consists in one of the most innovative part of the project. It is concerned with the development of a toolbox of formal methods for quantum programs developed in the language QuaML provided in WP1. Albeit almost inexistent as of now, formal methods are of uttermost importance in quantum computation. Indeed, the difficulty and the cost of testing and debug quantum programs make the development of methods to analyse and check a piece of code ahead of running it absolutely necessary. If the target is the compilation toolchain of the project, most of the techniques presented in the subsequent tasks are generic and can be used for other quantum toolchains.

Task 1. High-level code specification and verification

A QuaML program is at its core the description of a dynamic circuit. It contains both low-level and high-level circuit constructors. Low-level circuit constructors are such as quantum register allocations, unitary operations, measures, and quantum registers de-allocations, while high-level operations are circuit controlling, inversion, repetition, *etc*. In this task we focus on the specification and verification of the code seen as circuit-description language with low- and high-level operations.

Objectives. A language for formal specification of safety checks and contracts in QuaML developed in WP1. We want to be able to characterize both the well-formedness and the soundness of the generated circuits and ZX terms.
Methodology. We plan to rely on the use of Why3 and the expertise of CEA for this task.
Measure of success. Complete safety check of circuit generation done in quantum programs.
Risk management. This task rely on the Why3 formalization of D1.3, but also on a formal analysis on paper to characterize and define properly the situation. This paper analysis does not require anything.
Deliverable. D2.1 - T0+9 → T0+36 - Report and open-source code - Specification language for quantum algorithms.

Task 2. Abstract interpretation of entanglement

Entanglement is the most elusive resource of quantum computation. It is both essential for the quantum speed-up and yet a burden as it renders exponentially expensive the emulation of quantum computation on conventional machines.

Objectives. To analyze the evolution of entanglement during a computation: (1) by analyzing concrete problems we expect theoretical insight on the role of entanglement in quantum algorithms; (2) we expect practical outcomes that can be capitalized on in WP4 as the absence of entanglement between two parts of the memory eases the emulation on a classical computer.
Methodology. Abstract interpretation. Develop the proof of concept that have been introduced by a member of the consortium [**P08**]. In a first step, we plan to develop the method for a toy-language, and then expand it progressively to integrate it in the toolchain.
Risk management. The incremental approach mean that any delay in WP1 can be leveraged. Moreover, the failure of this task should not impair the project as it is not a bottleneck for anything.
Measure of success. Feasibility and accuracy of the analysis.
Deliverable. D2.2 - T0+0 → T0+36 - Report - An analysis tool for entanglement

Task 3. Complexity analysis through types

One of the selling point of quantum computation is the possibility to design algorithms solving classical problems faster than algorithms using only conventional machines. The speedup is characterized by the complexity of the algorithms: one problem is therefore to measure the complexity of a given algorithm.

Objectives. Design methods to distill informations on the complexity in space and time by static analysis of a given quantum program.
Methodology. Build on tools developed for “Implicit Computational Complexity” analysis. They consist in type systems such as Soft Linear Logic [L04], Tier-based typing [**LM13**,**HP15**], or Dual Light Affine Logic [BT04], for which well-typed programs are guaranteed to belong to a given complexity class. We plan to build on [DMZ10] to upperbound the time and space usage of quantum programs written in QuaML
Risk management. The approach can first be implemented on a toy-language before moving to the complete language. This is not impacted by any delay in WP1. Moreover, the failure of this task should not impair the project as it is not a bottleneck for anything.
Measure of success. Feasibility, efficiency and accuracy of the analysis.
Deliverable. D2.3 - T0+6 → T0+48 - Report - Implicit Computational Complexity for quantum programs

Task 4. Automatic logical resource estimation

From the perspective of the implementation, an essential factor is the actual number of resources needed by a particular quantum program: how many gates are required, what are the depths of the circuits, how many qubits are going to be used?

Objective. Automate the logical resource analysis for general quantum programs.
Methodology. We plan on type annotations in the spirit of Bounded Linear Logic [GSS92], since a QuaML program is essentially manipulating lists of qubits. Lists of qubits will be annotated with their size, and list combinators into functions on natural numbers. Preliminary works from one of the investigators show promising results.
Measure of success. The capacity to produce accurate resource estimates for the library of existing programs in Quipper.
Risk management. The approach can first be implemented on a toy-language before moving to the complete language. This is not impacted by any delay in WP1. Also: the failure of this task should not impair the project as it is not a bottleneck for anything. Finally, this can be seen as an alternative approach to D2.3, distributing the total risk of failure for resource analysis.
Deliverables. D2.4 - T0+9 → T0+30 - Report and open source code - Automated logical resource estimation.

WP3: Intermediate language for code optimisation

WP leader: LORIA (Jeandel) **WP collaborators:** LORIA, LRI

This work-package is the middle layer of our envisioned quantum compilation toolchain: it sits between the high-level language of WP1 and the backend of WP4. We choose the ZX-calculus [CD09] as intermediate representation (IR), target of the high-level language. ZX calculus is a powerful graphical language which captures fundamental properties of quantum mechanics like entanglement, causality and how they interact. This is an ideal IR, with the appropriate level of abstraction for code optimisation: ZX-calculus is abstract enough to have a simple and well-defined semantics. It is made so that low-level optimizations can be designed and checked correct against the semantics. It is yet low-level enough so that it can easily be compiled down to the hardware. In our case, the hardware is thought of as an emulation platform, based on a particular computational model: the QRAM model, or the MBQC model. For both of them the ZX-calculus comes with a natural mapping. However, in order to serve as a successful IR for our compilation toolchain, several aspects need to be worked out: each task attacks one of them.

Task 1. Towards a *dynamic ZX*

In their most general form QuaML programs generate families of dynamic circuits. Whereas

ZX-calculus is expressive enough to represent static un-parameterized circuits, the necessary dynamical behavior is a challenge in the integration of the ZX-calculus in the toolchain.
Objective. Extend the ZX calculus to represent the full range of quantum programs.
Methodology. We plan to address this obstacle by developing a "dynamic ZX". We plan to generalise the weakly parameterized !-graphs [KMS12] and use event structures [W80] that have recently successfully been used for modelling probabilistic calculi [CCPW17].
Measure of success. Development of a sound dynamic ZX-calculus. Quantum switches [**CAPV13**,**FP15**] will also a good testbench of success: while not possible to implement with static circuits, quantum switches can be implemented easily with dynamic circuits. A dynamic ZX calculus would also work for the same reasons.
Risk management. This task tackles the theoretical aspect of the problem, while the strongly related D1.4 is concerned with the implementation. If we fail to obtain a fully universal dynamic ZX-calculus, a fallback solution consists in staying with static ZX -terms.
Deliverables. D3.1 - T0+0 → T0+12 - Report - A dynamic ZX-calculus.

Task 2. Completeness and equivalence of code

One of the potential benefit to use an intermediate language is to be able to decide whether two programs are equivalent or not. To this end, one can transform two Quipper programs into ZX-diagrams and then decide whether one ZX-diagram can be transformed into the other or not.

Objective. Decision procedure for equivalence of programs using rewriting of ZX-terms.
Methodology. The decision procedure requires the completeness of the ZX-calculus, which is an open question [**PW16**]. We will address the question of the completeness by successive refinements. First, we will identify new needed equations by analysing concrete examples of quantum programs. To attack completeness of the language we plan to investigate normal form techniques, which will be turned into decision procedure for deciding equivalence of codes.
Measure of success. Ability to prove completeness or incompleteness result; size and elegance of the set of axioms. Efficiency of the procedure to decide equivalence of programs.
Risk management. Despite of our expertise on the question, proving the completeness of ZX-calculus is a risky task. If we fail to prove completeness, the approach can nonetheless be used but false negative may occur. This task is not prerequisite to any other.
Deliverables.
D3.2a - T0+0 → T0+12 - Open database - Lists of necessary axioms for the language.
D3.2b - T0+12 → T0+48 - Report - Completeness of the ZX-calculus.

Task 3: Code optimization

The rules of the ZX-calculus allows one to transform a ZX-term to optimize it. There are several possible objectives of optimization: depth, number of operations … The optimisation is a priori model driven: minimize the depth of a circuit or an MBQC is not the same, classical oracles is another examples.

Objective. Optimisation procedures of ZX-terms for the QRAM and the MBQC paradigms.
Methodology. Development of rewriting strategies. Investigation of optimization techniques based on back and forth translations between concrete models (e.g. equivalence between circuits and MBQC) [PDK15] can be considered in the unifying framework of the intermediate language.
Measure of success. Efficiency and accuracy of the optimisation will be a measure of success of code optimisation.
Risk management. First, the task is not prerequisite to any other. Then, if it depends on D3.2, it can very well be developed with only D3.2a available.
Deliverables.
D3.3 - T0+12 → T0+24 - Report, open-source code - Rewriting strategies for optimizations.

Task 4: ZX-calculus in practice.

To integrate the intermediate language to the compilation toolchain, a concrete implementation of the ZX-calculus is necessary.

Objective and methodology. Use the software Quantomatic to implement D3.3: it consists in implementing methods in Isabelle/ML. Success is measured in the capacity to implement all the optimizations of D3.3.
Risk Management. Quantomatic is an open source software is mainly developed in Isabelle, and LRI has a strong expertise in Isabelle [**BBF+16**]. One of the main challenges is the extension of Quantomatic to deal with dynamic ZX-terms.
Deliverable. D3.4 - T0+24 → T0+48 - Open-source code - Optimization driver for the toolchain

WP4: Back-end and emulation on classical super-computer

WP leader: BULL (Allouche) **WP collaborators:** LRI,LORIA

General principles of HPC simulation. Simulation of quantum computations is very costly, both in CPU and memory. Only HPC techniques (software & hardware) can enable the emulation of non trivial programs. This is what this WP will focus on.

It is also noteworthy that, while besides the scope of this WP, HPC plays an important role for the research on hybrid architectures [FRL16]. Indeed, in future hybrid quantum/HPC architectures, the quantum computing power will be devoted to "hard" problems, i.e., circuits that cannot be simulated in polynomial time. Hence, efficient HPC simulation techniques enable to identify these "hard" circuits.

As always in HPC, the 2 aspects have to worked in order to optimize the execution:
- algorithmics and software implementation
- computing platform, and in particular usage of hardware accelerators

This work package will consist in providing optimized translation and execution engines for code generated from QuaML and the ZX calculus intermediate layer. Optimization criteria will be a tradeoff or combination between execution time and program capacity (number of qubits).
Consequently, we will investigate the best strategies for both algorithmic and computing architecture simulations, for a list of reference quantum programs that will decided at the beginning of the project. It seems clear from the very first analysis that the efficiency of our approach will depend on the quantum programs being run. It will be part of the project to perform this analysis in deeper details, with a performance benchmarking method. If we shall consider the usual QRAM and circuit model, for simulation we shall also investigate the specific emulation of MBQC computation [NHZS15] that may be better suited for emulation of ZX-terms. For these simulations, the interactions between classical and quantum computing parts will be studied, with the perspective of future heterogeneous models for quantum computer architectures.

Overall objective of WP. This work-package consists of tasks in sequence, aiming at a working simulation platform with rationales for the choices being made at the end of the project.
Methodology. The tasks in this work-package are essentially streamlined and move from a generic implementation to the incorporation of the optimizations developed in WP3 within the emulation framework.
Risk management. There are no inherent risks in the development of the simulation platform *per se*: this axis of research is backed up by Atos-Bull, and proceed with standard code development procedure. The question is whether and how we'll manage to beat the existing emulation platforms and capitalize on our novel ZX approach. Finally, as for the other WPs, we do not have to wait for WP1 to complete: to get early testbenches we will start ahead with developing an output driver for Quipper.

Task 1 : Integration with Bull's quantum simulators and experimentation

Integration of QuaML and ZX-calculus outputs into Bull's quantum simulators. This will allow to start working with a first runnable backend, which will provide a first reference.
This task will consist in developing the integration code, and then do a first performance ref.
Deliverable. D4.1 - T0+0 → T0+9 - Proprietary source-code - interface to Bull's quantum simulators.

Task 2 : Definition of optimal criteria and benchmark reference

- Collecting the reference quantum programs, on which the benchmarks will be run: linked with

D1.1

- Definition of the HW reference platforms - the benchmark will be run within these references. In particular, it will be important to discuss whether or not including these: SMP server, HPC server, FPGA board, GPU blades.

Deliverable. D4.2 - T0+9 → T0+18 - Open database - Bank of reference quantum programs for benchmarks and selection of targeted simulation platforms.

Task 3 : Characterization of the impact of the code optimizer on the simulation performance - Identification of specific implementations

This task will study the influence of the code optimizer developed in WP3 on the performance of the simulation, for all the programs and all the HPC target architectures. A fine-grained profiling will exhibit either the right influence of optimization strategies, or the bottlenecks in the simulator, that prevent the optimization to be efficient. Consequently, this task will identify:

- the optimizing strategies that pay, at least for a simulator backend
- the specific optimizations the simulation backend should implement in order to take more benefit from the optimized compile chain

Deliverable. D4.3 - T0+18 → T0+36 - Public report and open-source code - Profiling of optimization strategies

Task 4 : development of specifics

This task will consist in developing the specific optimizations in Bull's simulators, and optimize them, on the identified platforms. It is likely not all target platforms will benefit from the same optimizations, and hence a tradeoff will have to be decided, as privileging the most promising platform(s).
Deliverable. D4.4 - T0+36 → T0+42 - Public report and proprietary source-code - Optimizations specific to Bull simulators

Task 5: benchmarking, analysis and refinement

This task will consist in benchmarking, profiling, optimizing the new version of the simulator backend, which embeds the specifics.
Deliverable. D4.5 - T0+42 → T0+48 - Final (public) report of WP4

WP5: Dissemination

WP leader: LRI (Valiron) **WP collaborators:** All

WP5 focuses on the dissemination of the results obtained during the project. Moreover, in order to encourage other solver developers to focus on problems that matter for program verification, we will make the benchmarks produced during WP1 readily available. Last but not least, we will present the theoretical work done during WP2 and WP3 at conferences or in journals.

- *Publications*. We will produce early technical reports that could be later turned into publications.
  Website. We will provide from the project website an online IDE to the various tools developed, and an API to Bull's simulator will eventually be provided. The website will also be equipped with MOOCs and tutorials on quantum programming.
- *Workshop.* We also plan a 2- or 3-day open workshop on the third year in order to review progress and difficulties. This seminar will be an opportunity to invite a few external experts.
  Open source. Most framework and theories implemented during this project and aimed for publication will be so with an open-source license. Tutorials and a mailing-list will be provided for a community to form.
- *Benchmarks*. We will publish our benchmarks and make them available from the website of the project, to serve as future references for the community.

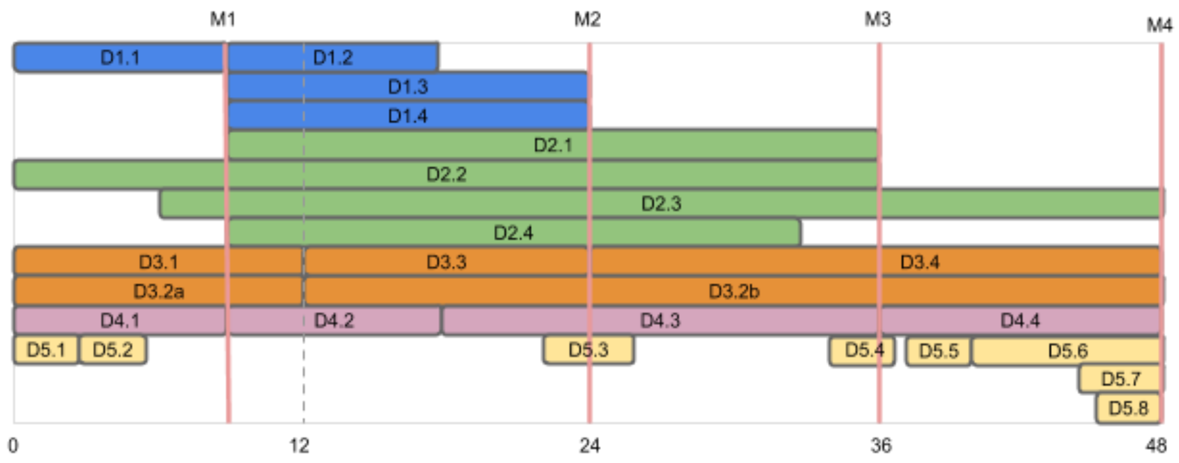
Deliverables.
D5.1 - T0+3 - Project website, mailing lists, code repositories
D5.2 - T0+6 - Tutorials on quantum programming on the website
D5.3 - T0+28 - Set up of online IDE and MOOC.

D5.4 - T0+36 - Open workshop
D5.5 - T0+40 - Update on tutorials with quantum formal methods
D5.6 - T0+42 → T0+48 - Contribution to the Quantomatic by integration of our development.
D5.7 - T0+48 - Public white paper on SoftQPro
D5.8 - T0+48 - Report on dissemination activities

Deliverables recap (colors correspond to WPs)**:**

D1.1	T0+0→T0+9	**Report** - Formal description of the language
D1.2	T0+9 → T0+18	**Report** - Interfaces for the formal language
D1.3	T0+9 → T0+24	**Report** & **open-source code** - incorporation of Quipper’s libraries within the formal language QuaML
D1.4	T0+9 → T0+24	**Open-source code** - implementation of the formal language QuaML
D2.1	T0+9 → T0+36	**Report** & **open-source code** - Specification language for quantum algorithms
D2.2	T0+0 → T0+36	**Report** - An analysis tool for entanglement
D2.3	T0+6 → T0+48	**Report** - Implicit Computational Complexity for quantum programs
D2.4	T0+9 → T0+30	**Report** & **open source code** - Automated logical resource estimation.
D3.1	T0+0 → T0+12	**Report** - A dynamic ZX-calculus
D3.2a	T0+0 → T0+12	**Open database** - Lists of necessary axioms for the ZX-calculus
D3.2b	T0+12 → T0+48	**Report** - Completeness of the ZX-calculus
D3.3	T0+12 → T0+24	**Report** & **open-source code** - Rewriting strategies for optimizations
D3.4	T0+24 → T0+48	**Open-source code** - Optimization driver for the toolchain.
D4.1	T0+0 → T0+9	**Proprietary source-code** - Interface to Bull’s quantum simulators
D4.2	T0+9 → T0+18	**Open database** - Bank of reference quantum programs.
D4.3	T0+18 → T0+36	**Public report** & **open-source code** - Profiling of optimization strategies
D4.4	T0+36 → T0+42	**Public report** & **proprietary source-code** - Optimizations specific to Bull simulators
D4.5	T0+42 → T0+48	**Public report** - Benchmarking, analysis and refinement.
D5.1	T0+3	Project website, mailing lists, code repositories
D5.2	T0+6	Tutorials on quantum programming on the website
D5.3	T0+28	Set up of online IDE and MOOC.
D5.4	T0+36	Open workshop
D5.5	T0+40	Update on tutorials with quantum formal methods
D5.6	T0+42 → T0+48	Open-source code - Integration of our development to Quantomatic
D5.7	T0+48	Public white paper on SoftQPro
D5.8	T0+48	Report on dissemination activities
Milestones:		
M1	T0+9	*Formal description of the language QuaML, specifications for the formal methods.*
M2	T0+24	*Open-source code of an implementation of the formal language QuaML*
M3	T0+36	*Open workshop; Running specification, analysis and certification methods.*
M4	T0+48	*Public white paper on SoftQPro; Project overall review and follow-up.*

Gantt chart:



Related projects and funding. The members of the consortium have no previous or ongoing projects and funding received in connection with this proposal.

III. Impact and benefits of the project

Fields of Impact. Quantum Computing (QC) is today the most promising track to supplement Moore's law, whose end is expected with the engraving at 7 nm, in less than 5 years. Thanks to the exponential calculation power it will bring, it will represent a decisive competitive advantage for the companies and the countries that will control it, in the computer market, and more particularly on the HPC segment, which is well known as key for both industrial innovation and scientific excellence.

Quantum Computing is also a major security issue, since it allows to break today's asymmetric cryptography. Hence, master of QC is also of higher importance for National Security concerns. Recent scientific and technical advances suggest that the construction of the first quantum computers will be possible in a the coming years, even if their first capacity will not allow to reach the so-called quantum supremacy at first.

As a result, the major US players in the IT industry have embarked on a dramatic race, putting huge resources: IBM, Microsoft, Google and Intel each invested between 20 and 50 million euros, and are putting strong means to attract and hire the best scientists of the planet. Some states have launched ambitious national programs, including Great Britain, the Netherlands, Canada, Australia, Singapore, and very recently Europe, with the to come 10-year FET Flagship program in Quantum Engineering.

If a large part of these means goes to R & D in quantum hardware, there is yet an important need and real opportunities for leadership in the field of quantum software, which will be quite disruptive to the software industry. As a consequence, the economic development of quantum software should explode with the arrival of the first quantum processors, following the same model as the conventional software. It is noteworthy that three IT majors (namely IBM, Microsoft, Google) have started to "capture" this very early market of quantum software. As an example, IBM claims 40.000 users.

National initiatives are still possible and highly desirable, in order to provide complete alternatives and differentiating build blocks. The SoftQPro project takes place in this scientific and economic challenge, and directly addresses several of the core scientific and technological problems of quantum software rogramming languages, Compiling chain, Formal Verification, and Experimentation environment, through HPC simulation. We propose innovative, differentiating, features, as compared to the existing solutions we mentioned.

The outputs of the SoftQPro project aim at shaping and federating the community around the novel approach we offer.

- *Theoretical outputs*. The SoftQPro project is expected to produce rank-A publications in journals and workshops, presenting the techniques that were developed. The website will serve as a repository for public tutorials on quantum programming, ZX-calculus and quantum compilation.

- *Concrete outputs.* The SoftQPro project also aims at generating prototypes, both open-source for the non-specifics and proprietary for the code related to Bull's equipment. Along the course of the project, benchmarks in open-data and public white papers will be produced, setting milestones for the design of a sound compilation toolchain.

Dissemination. Our deliverables are thought to be used by a rather *wide audience*: quantum algorithm researchers ; quantum software scientists ; students in quantum software ; end users software developers who want to "go quantum".

The examples of former collaborative projects in this same scope (mentioned earlier: Quipper, ScaffCC,...), as well as commercial projects (Microsoft Liquid, IBM Quantum Experience,...), let us think the dissemination will be easy among the well known community of quantum software researchers. As of the "end users" community, Atos-Bull is committed to disseminate these results through ***industrial events*** and dedicated communication, as it does for its own R&D program. As the simulation capability is key for the adoption by end-users, Bull commits to provide the partners a cloud-based access to the HPC simulation engine after the end of the project, which will be extensible to other educational & research partners, as well as end users, subject to acceptance of terms of use.

The academic partners will communicate their research outputs through ***high-ranking venues and invited talks***. The ***workshop*** we aim at organizing at T0+36 will be an incubator for fostering the community. An important aspect is communication towards graduate students and future engineers. We plan to set up in LRI and LORIA ***tutorials sessions and courses on quantum software techniques***, while proposing ***projects centered around quantum programming*** at the engineering school CentraleSupelec. Targeted towards quantum software scientists and end users software developers who want to "go quantum", but also quantum algorithms researchers, the ***website*** will contain an ***online IDE*** for the tools developed and series of ***tutorial with a MOOC*** on quantum programming.

Economic Outcomes. While not generating any revenue, the market of quantum software is, at least for the 3-5 coming years, *a market of influence and leadership*. Those who will demonstrate innovation and technological leadership will influence and federate the to-come industry, *with a "winner takes all" effect typical of the digital industry*. We strongly believe that SoftQPro will ***help the French quantum community to engage into that competition***, by providing valuable, steady grounds for innovative quantum programming frameworks, what is at the heart of the quantum software needs.

Meet the challenges of the Défi. Défi 7 and more generally the ANR work program, wishes to contribute to the European research construction. This project aims at building a strong leading team, gathering both academics and industrials, *ready for the upcoming Flagship on Quantum Technologies and QuantERA ERA-NET Cofund in Quantum Technologies* mentioned in the ANR work plan.
Envisioning the implementation of a quantum algorithm is not a simple task. The backend is still elusive, the sizes of quantum circuits for realistic input sizes are daunting, and there are as of yet very few tools to obtain guarantees on the run of the algorithm and on the results. The development of a compilation platform for quantum computation lies therefore well within the axis "software engineering" ***(Defi 7 – Axe 3 : Science et technologies logicielles***). The objectives of the project are aligned with the three main themes of the axis. The first theme is concerned with computing platforms. In the context of quantum computation, such platform are yet to be designed: one of the goal of the project (work-packages 2 and 3) is to capitalize on the features of each backend to get the best results. The second theme focuses on methods and design tools for software: this is the realm of work-package 1, devoted to the development of the quantum programming language QuaML. Work-package 1, together with work-package 2 tackles the question asked in the last theme: the validation of software. We aim at developing a toolbox for the certification and the verification of properties of quantum programs.
If the core of the work fits within the 3rd axis of Defi 7, the project also reaches two other axes. Indeed, the development of this compilation tool-chain will require the use of theoretical apparatus: implicit computational complexity, type theory, abstract interpretation, … leading to ***Axis 1 (Informatique théorique)***. The CEA, LRI and LORIA partners are strong in this aspect. Also, because one of the back-end we have in mind is the emulation of quantum computation, this falls in the realm of ***Axis 6 (Simulation numérique)***. This aspect is backed up by BULL and LRI.