D5.1 – Architecture of the ComPat System

Due Date	M9
Delivery	M9
Lead Partner	PSNC
Dissemination Level	Public
Status	Submitted
Approved	Executive Board: yes
Version	V1.0

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 671564.

DOCUMENT INFO

Date and version number	Author	Comments
19.05.2016 v0.10	Tomasz Piontek	First version
25.05.2016 v0.20	Bartosz Bosak	Executive summary, ComPat Architecture section and diagram
06.06.2016 v0.30	Keeran Brabazon Dirk Schubert	Allinea Tools
08.06.2016 v0.40	Alfons Hoekstra Saad Alowayyed	Computing Patterns, Performance Prediction Models
08.06.2016 v0.50	Derek Groen	FabSim, MPWide
10.06.2016 v0.55	Piotr Kopta	QCG Middleware
10.06.2016 v0.60	Peter Coveney	ComPat Applications
13.06.2016 v0.70	Stephan Hachinger	Experimental Execution Environment
13.06.2016 v0.80	Arjen van Elteren	AMUSE
14.06.2016 v0.90	Ulf Schiller	Remarks and corrections
17.06.2016 v0.95	Bartosz Bosak Tomasz Piontek	Version for the internal review
23.06.2016	James Suter Helmut Heller	Comments and remarks on v0.95 (internal review)
27.06.2016 v1.0	Tomasz Piontek	Final version

CONTRIBUTORS

The contributors to this deliverable are:

Contributor	Role
Tomasz Piontek	Technical Leader, WP5 Leader and author of this deliverable
Bartosz Bosak	WP5 contributor
Piotr Kopta	WP5 contributor
Keeran Brabazon	WP4 contributor
Dirk Schubert	WP4 contributor, WP4 Leader
Alfons Hoekstra	WP2/WP3 Contributor, Project Coordinator
Saad Alowayyed	WP3 contributor
Peter Coveney	WP3 contributor
Stephan Hachinger	WP6 contributor
Ulf Schiller	WP3 contributor
Arjen van Elteren	WP2 contributor

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

AMUSE	Astrophysical Multipurpose Software Environment
CUDA	Compute Unified Device Architecture
ECOL	Energy Consumption Optimization Library
ECOS	Energy Consumption Optimization Service
EEE	Experimental Execution Environment [Project Testbed]
ES	Extreme Scaling [Pattern]
GPGPU	General-Purpose Graphics Processing Units
HMC	Heterogeneous Multiscale Computing [Pattern]
HMM	Heterogeneous Multiscale Manager
HPC	High Performance Computing
HPMC	High Performance Multiscale Computing
ISR	In-Stent Restenosis
MAPPER	Multiscale Applications on European e-Infrastructures [Project]
MML	Multiscale Description Language
MPI	Message Passing Interface
MTO	Multisite Transport Overlay
MUSCLE	Multiscale Coupling Library and Environment
PRACE	Partnership for Advanced Computing in Europe
QCG	Quality in Cloud and Grid
RC	Replica Computing [Pattern]
XMPP	Extensible Messaging and Presence Protocol

1 Executive summary

The aim of this document is to provide an overview of the ComPat system architecture. In this document, we present the established technology stack and discuss the integration aspects of all parts of the ecosystem for building and executing multiscale applications based on the concept of High Performance Multiscale Computing (HPMC) patters. The ComPat architecture has been designed in close collaboration with Work Packages 2 (Patterns), 3 (Applications), 4 (Tools) and 6 (Experimental Execution Environment, EEE). Based on our in-depth analysis, supported by questionnaires completed by the relevant Work Packages and following the generally accepted methodology of software design and development, we decided to divide the architecture into several logical layers, namely high-level tools, applications, computing patterns, middleware and execution environment. These layers, their components, as well as the relations between them, are the main subject of this document.

The ComPat project distinguishes between fast track and deep track components. Therefore, this document also presents an internal schedule for developing new components and describes how their functionalities are related to the project's objectives. The components assigned to the fast track are designed to be deployed in the first phase of the project to enable basic functionality of the overall system, while the deep-track components will be deployed subsequently to support more sophisticated scenarios. The assignment of components into the deep track is only possible if the implementation and deployment of such components can be scheduled for later in the project without causing problems with dependencies between tasks in the Work Packages.

The ComPat project builds upon the achievements of earlier developments wherever possible and justified. Thus, in addition to completely new software that will be implemented during the project, a number of previous or complementary software components are also incorporated into the architecture. This document provides a clear distinction between new ComPat components and those which have been developed externally. For the existing components the required adaptation effort is described.

This deliverable presents the starting point for all subsequent technical developments in ComPat, and therefore represents an important early milestone in the project.

2 ComPat system architecture

It is not trivial for computing systems' designers to address the ever-changing requirements of leading-edge science. Advanced multiscale simulations, benefiting from the concept of HPMC patterns, have to be supported by a dedicated, flexible and well-defined system. In this document, we present the design of the ComPat environment and describe how the environment addresses these requirements. The information provided here discusses the basics of integration between the main ComPat layers, i.e. high-level tools, applications, computing patterns, middleware and execution environment, and, as such, this deliverable will be an essential design document for all the technical Work Packages. By preparing this

document in the first phase of the project, we align the work and roles of the project participants as early as possible.

In order to develop an architecture that fully meets the requirements of applications on the one hand, and takes all circumstances of the underlying HPC infrastructure into account on the other, the work on the architecture design was started by tabling two relevant questionnaires (Annex A, B) to the relevant ComPat technical Work Packages. The first questionnaire was targeted at groups involved in application design (Work Packages 2 and 3). The target audience of the second questionnaire were resource providers represented by Work Package 6. The aim of these two questionnaires was to identify, in advance, all the required functionality foreseen by applications and to determine all the procedural, administrative and technological limitations in accessing resources. Using the received feedback, we were able to propose a complete and stable system architecture, which is presented below. Nevertheless, taking into account the explorative and emerging character of the project, it must be noted that the proposed architecture may be slightly altered during the next stages of the project. However, we do not expect any significant changes to its core design.

2.1 General view on the ComPat architecture

Since ComPat is an application driven project, the successful creation and execution of our grand challenge applications is of the utmost importance for the success of the project, as well as facilitating a qualitative verification of the (potential) impact of ComPat innovations. This fact was taken into account when the ComPat consortium defined three general multiscale computing patterns (Extreme Scaling, Heterogeneous Multiscale Computing and Replica Computing) as a generic layer between the applications and the (emerging) exascale computational environment. The main aim of the construction of such patterns (the so called HPMC patterns) is to simplify the implementation of HPC multiscale applications and to variously enhance their execution. From the point of view of the applications, the patterns determine the ordering and composition of the single scale models that are coupled within a multiscale application. It is foreseen that the patterns, by providing well-defined functionalities for common requirements of multiscale scenarios, e.g. fault tolerance or support for energy-aware execution, will allow to reduce the “time to market” and enhance the applications’ stability and performance. We further discuss this, and provide examples from our grand challenge applications, in Section 2.3.

In order to orchestrate the execution of the pattern-based application on HPC resources and specifically exascale systems, we require a comprehensive middle- and low-level technology stack. The architecture of the ComPat system has to support and correspond to the typical scheme of development and execution of multiscale applications, as presented in Figure 1. This scheme is based on practices worked-out in earlier projects, particularly in MAPPER, and enriched with modules required for High Performance Multiscale Computing (most notably, Multiscale Computing Patterns, energy awareness, and debugging/profiling). We can distinguish here between the development of patterns, single-scale

application kernels and the combined development of entire multi-scale applications. As shown by the thick arrows, the process of development is organized into loops. The undoubtable advantage of the scheme is the possibility of working in parallel by different ComPat Work Packages.

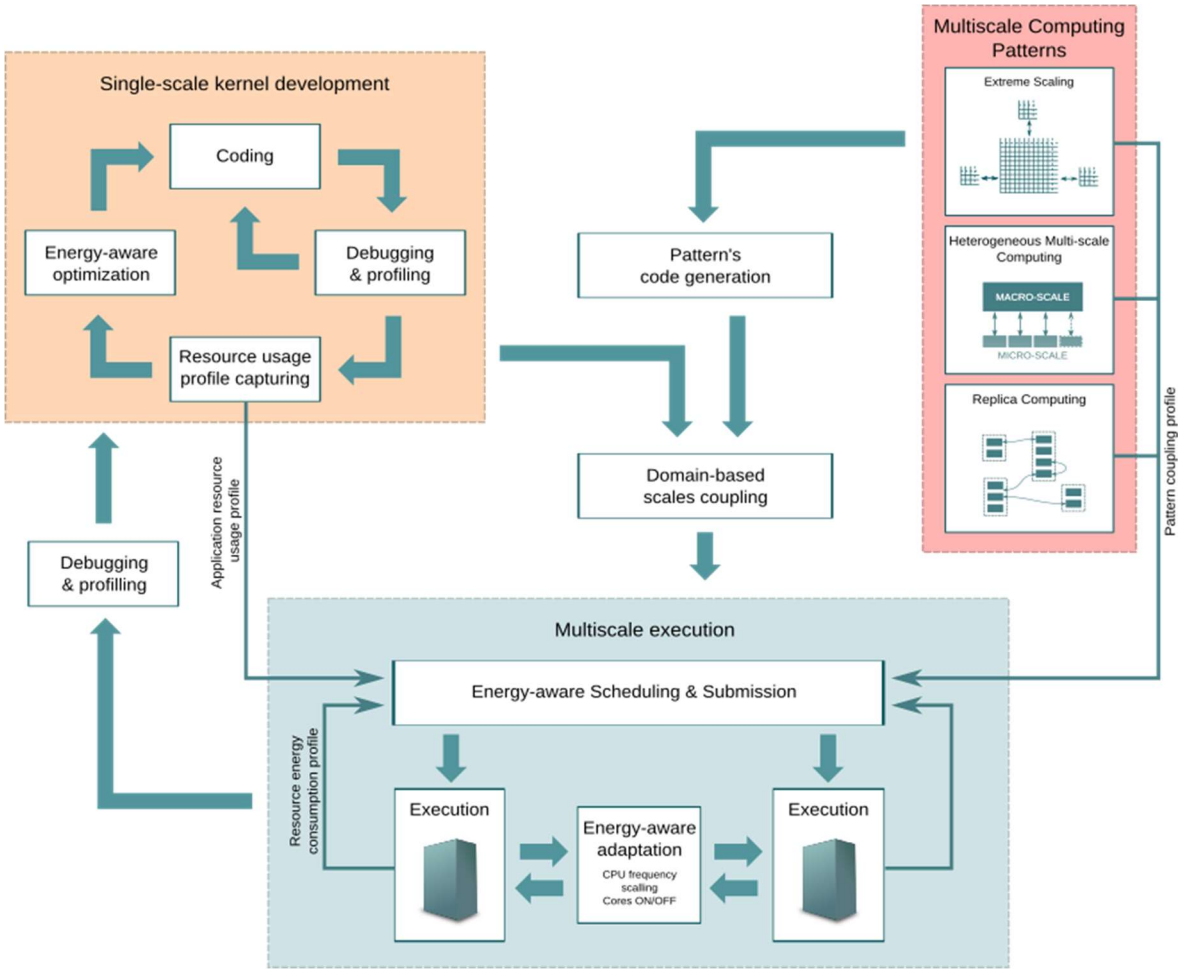

Figure 1 ComPat's multiscale application development and execution loop

Nevertheless, since the execution of a multiscale application on the EEE, together with its optimization, is an integral part of its development, it is necessary for the project to provide unified access to the execution environment as soon as possible. Therefore, it was decided to distinguish between fast-track and deep-track components. The components assigned to the fast track are required to be deployed in the first phase of the project, even with limited functionality, to enable basic operation of the overall system, while the group of the deep-track components will be deployed subsequently to support more sophisticated target scenarios. The assignment of components into fast-track and deep-track development is provided in Table 1.

In Figure 2 we present the general ComPat architecture. The ComPat technology stack is composed of tools, applications, computing patterns, middleware services, and the exascale multiscale simulation framework, fully operational on an Experimental Execution Environment, consisting of HPC/PRACE resources throughout Europe. Whenever possible, the ComPat technology stack benefits from the

existing software components and achievements of previous or complementary projects. Thus beside the completely new developments needed to meet the specific demands of the multiscale computing patterns and/or the requirements and demands related to exascale performance, there are also mature components incorporated into the ComPat architecture that with or without modification compose the whole system. A summary presenting all components on the basis of their origin and required development effort within the ComPat project is shown in Table 2.

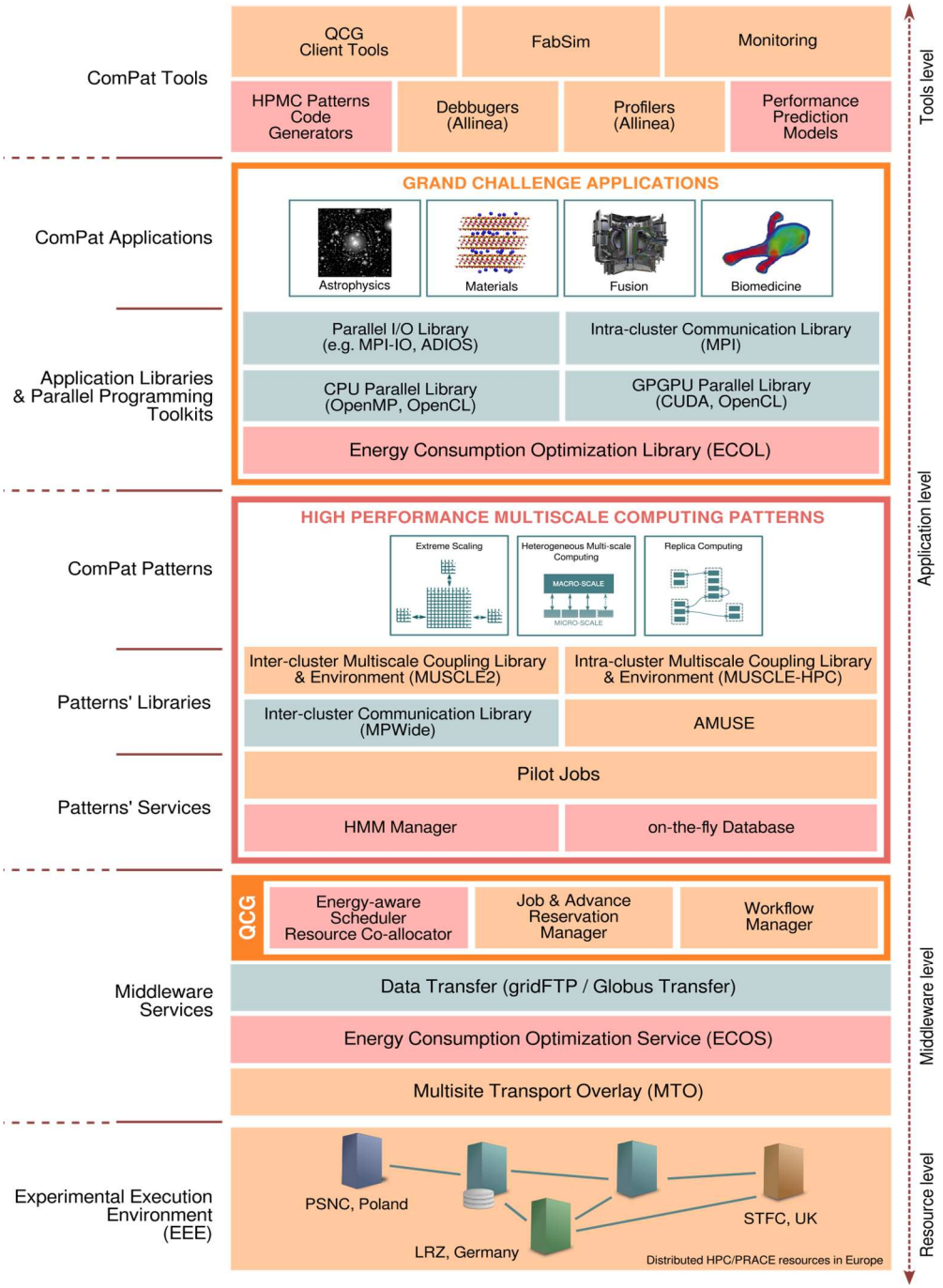

Figure 2 General ComPat architecture. New components are in pink, the existing components that will be modified in some way are in orange, and the existing components that will not be altered in any way are in grey.

Layer	Fast Track	Deep Track
Tools	QCG-Client Tools (QCG-Client) Single-scale Debuggers & Profilers FabSim	QCG-Client Tools (QCG-Portal, QCG-Now) Monitoring Multi-scale Debuggers & Profilers
Applications	2 selected applications, to be decided	6+ applications
Application libraries and toolkits	MPI, OpenMP, CUDA, OpenCL	ADIOS, ECOL
Patterns	ES RC	HMC
Patterns' libraries	MUSCLE MUSCLE-HPC	MPWide AMUSE
Patterns' services		Pilot Jobs HMM Manager On-the-fly Database
Middleware	Job & Advance reservation manager (QCG-Computing) Scheduler and resource co-allocator (QCG-Broker) GridFTP MTO	Workflow manager Energy-aware scheduler and resource co-allocator ECOS
EEE	3 selected resources	5+ resources

Table 1 Division of ComPat components into fast-track and deep-track groups

	Component	Origin	Development effort in ComPat
Tools	QCG-Client tools	External (QCG)	Adaptation to the requirements of ComPat patterns
	FabSim	External	Integration with other ComPat tools and adaptation to support the requirements of the patterns. On-demand adoptions by Brunel University
	Monitoring	External (QCG)	Support for new application types
	HPMC Code Generators	ComPat	Development from scratch
	Debuggers	External (Allinea)	Extensions to support debugging of multi-scale simulations
	Profilers	External (Allinea)	Extensions as well as new development to support profiling of multi-scale simulations
	Performance Prediction Models	ComPat	Development from scratch
Applications	ISR	External (UvA)	Support for hybrid ES/RC patterns, porting to EEE machines, performance monitoring and modelling
	RBC and Platelet transport	External (UvA)	Development of MUSCLE wrappers and scale bridging, porting to EEE

			machines, performance monitoring and modelling
	Blood Rheology	External (UvA)	Development of MUSCLE wrappers and scale bridging, porting to EEE machines, performance monitoring and modelling, development and testing of required HMM database functionality.
	Binding Affinities	External (UCL)	Support for RC jobs, including testing of Pilot Jobs and porting to EE machines.
	Nanomaterials	External (UCL)	Development of scale bridging methods (model building / parameterisation etc) support for HMM database functionality
	Aneurysm flow dynamics	External (UCL)	Development of MUSCLE wrappers for coupling high-to-low resolution simulations as part of Extreme Scaling pattern, performance monitoring and modelling, decomposition based time-to-solution prediction.
	Fusion	External (IPP)	Development of MUSCLE wrappers and scale bridging methods, porting, performance monitoring and scenario optimization on EEE machines.
	Astrophysics	External (Leiden)	Integration of performance monitoring and modelling with AMUSE and MPWide.
Applications' libraries	Parallel I/O Library (e.g. MPI-IO, ADIOS)	External	-
	MPI (e.g. OpenMPI)	External	-
	OpenMP	External	-
	CUDA	External	-
	OpenCL	External	-
	ECOL	ComPat	Development from scratch
Patterns	ES	ComPat	Development from scratch
	HMM	ComPat	Development from scratch
	RC	ComPat	Development from scratch
Patterns' libraries	MUSCLE2	External (MAPPER)	porting and testing on EEE, merging with MUSCLE-HPC, extending with dynamic instantiations of single scale models
	MUSCLE-HPC	External (University of Geneva)	Merging with MUSCLE2
	MPWide	External (MAPPER)	-
	AMUSE	External (University Leiden)	Support for HPMC Patterns
Pattern s' service	Pilot jobs	External	Support for specific HPMC Patterns needs
	HMM Manager	ComPat	development from scratch

	On-the-fly Database	ComPat	development from scratch
Middleware services	Energy aware scheduler & resource co-allocator	External (QCG)	Incorporating energy-metrics into scheduling process Support for patterns-based applications
	Job & advance reservation manager	External (QCG)	Support for patterns-based applications
	Workflow manager	External (QCG)	Support for patterns-based applications, Fault tolerance issues
	Data transfer	External	-
	ECOS	ComPat	Development from scratch
	MTO	External (MAPPER)	Possible rewrite to support new scenarios
EEE	Several resources of peta-scale class	External (PRACE resources)	Configuration supporting multi-site execution of ComPat jobs

Table 2 The list of ComPat software components with respect to their origin and development effort needed to incorporate them into the ComPat stack. The same colour scheme is used as in Fig. 2.

In the following sections we discuss all the layers of the ComPat architecture one by one. A special focus is given to relations between elementary components and their role in the system.

2.2 Tools

The aim of the ComPat tools is to offer wide support for users in the process of development and execution of multiscale applications. We define the following generic groups of tools:

- application submission and monitoring tools,
- pattern-based code and/or input files generators,
- application debuggers and profilers,
- performance prediction models.

In the following subsections we present all of these groups.

2.2.1 Application submission and monitoring tools

To execute jobs on the ComPat resources several submission and monitoring tools are offered. At this stage of the project, the scope of functionality provided by the selected tools is sufficient for execution of basic multiscale jobs. This was validated in a previous project (MAPPER). However, we anticipate a number of extensions to the tools and formal description languages in order to fulfil the specific requirements for High Performance Multiscale Computing, i.e. support for extensions in the Multiscale Modelling Language (MML) description to handle patterns and energy-efficiency constraints, fault tolerance in exascale environments or flexible monitoring of the progress of running application.

2.2.1.1 QCG submission tools

The underlying QCG system has a set of dedicated access tools, including command-line, desktop and portal-based versions. We anticipate that the command-line QCG-Client, together with the supporting FabSim tool described in the next section, will be the primary access tool for the whole project. Taking into account preferences expressed by ComPat users as well as the complexity of selected multiscale applications, we do not expect, certainly in the first phase of the project, the need to use any graphical interfaces.

The QCG-Client, is a set of command-line tools, inspired by batch system commands, which allow a user to submit, control and monitor a large number of various types of grid jobs as well as to reserve resources to obtain a requested level of quality of service. Since the QCG-Client's user interface is analogous to common batch systems' interfaces, the effort needed to start using the tool is relatively small. To be processed by QCG, tasks have to be described in a formal way. The basic description format, called QCG-Simple, is preferred and sufficient for a majority of tasks. However, it does not yet allow users to describe more sophisticated scenarios like workflows or tightly-coupled multi-scale simulations. This format will be extended appropriately in the next phases of the project. For now, all advanced scenarios supported by the QCG middleware have to be described by using the native XML based format, called QCG-Profile, that also will be slightly extended to support project goals. It is worth noting that QCG-Client provides valuable support for interactive access to resources if only a resource exposes this capability. Thus, depending on their particular needs, users can easily submit an interactive task to a cluster and either run their command line applications in interactive mode or compile their own code and process some debugging sessions directly on computing nodes.

2.2.1.2 FabSim

FabSim [1] is a Python-based automation toolkit for scientific simulation and data processing workflows. It enables users to perform remote tasks from a local command-line, and to run applications while curating the application data in a systematic manner. FabSim also contains a system for defining machine specific and application-specific configurations, with frequently used default settings being curated through a GitHub repository. FabSim is currently in use across several universities to automate computational tasks in fields such as bloodflow modelling, nanomaterials modelling, and protein-ligand binding calculations. In ComPat, we have already added support for GridFTP to FabSim and created an official software release. In the project we will use and further extend FabSim, incorporating the ability to flexibly instantiate complete computing patterns for several of the applications using one-line commands. FabSim will be integrated with other components of the ComPat software stack, particularly with QCG-Broker and coupling tools such as MUSCLE and/or MPWide.

2.2.1.3 Monitoring

Long-running and extremely parallelized simulations, like the exascale multiscale simulations we target, need to be effectively monitored during their execution on an HPC infrastructure in order to effectively plan experiments and to not waste resources. Therefore, we decided to provide monitoring tools which help users to check the correctness of jobs as well as to estimate the time needed for their completion. As a basic way of tracking the progress of running applications we propose notification messages with portions of the application output sent directly to the user as a mail or chat message. For those who need more advanced monitoring capabilities ComPat will provide a dedicated web portal built upon existing QCG middleware where the progress can be presented in the form of a set of tables, charts or diagrams according to the predefined template designed for a specific application or case.

2.2.1 Multiscale computing patterns code generators

The configuration and specification of multiscale models will be done in the existing Multiscale Modelling Language. However, the tools in ComPat will employ their own meta-code or configuration format. Based on existing tools such as the jMML library, configurations for high-level tools as well as middleware can be automatically generated. These can range from job descriptions, instructions for profiling a multiscale application, to a runtime topology for making scheduling decisions and performance models. Starting from an xMML description of the multiscale application, a task graph will be extracted and then mapped to generic multiscale computing pattern task graphs. Next, with information on the execution of the single scale model (coming from profilers, Section 2.2.3 and/or from performance models, Section 2.2.4) and given one of the three patterns, sufficient information will be generated for the QCG middleware components to be able to schedule and execute the multiscale application in the most efficient way. In the process of scheduling and brokering the system will take into account both users' preferences and constraints defined by the EEE resource owners.

2.2.2 Debuggers and profilers

The debugging tool Allinea DDT [2,3]will be extended to enable more versatile support for multiscale simulations and to be seamlessly integrated with the whole ComPat stack (mostly with coupling libraries and the middleware services). We can expect that debugging two (or more) closely coupled simulations may be required to understand an inter-dependent problem – and we will consider how this can be achieved through changes at the framework level and tool level. The Allinea MAP parallel profiler [4,5] and the Allinea Performance Reports [6,7] tools platform will be used within the project to obtain performance profiles of the applications, which can be used to provide input to the application developers. A performance profile contains information regarding an application run, providing information on the CPU, MPI and I/O activity, as well as providing more detailed information such as the use of vectorisation, multi-threaded performance, energy consumption over the run, and many more.

Furthermore, the profiling tools will be extended with the aim of providing support for each of the proposed computing patterns. The patterns require time and/or energy optimization - both within a single scale simulation and also when the scales are coupled. We will develop methods to merge the profiling data from multiple executions to provide information regarding the execution of the whole simulation, making the task of optimization easier for application developers. Initially we target simple couplings of multiscale applications but will target automatically creating such profiles based on the coupling frameworks developed in ComPat.

We will work to provide the tools through which performance can be profiled for the multiscale patterns. These tools can be used to obtain performance data from every application executing in a multiscale simulation. Such profiling would aggregate related performance data within multiscale simulations – providing the ability to determine how the performance of a multiscale simulation can be improved. As resources such as storage or networking of one simulation can impact the performance of others on many HPC systems, the performance profiling of the multiscale simulation is necessary. This data will also be used to validate the performance prediction models generated as part of Task 4.5.

The use of Allinea tools will enable analysis from a range of perspectives, including:

- Performance prediction: The existence of measured performance data provides insight into the variability of individual simulations, particularly those categorized as parameter sweeping tasks of the same single scale simulation. Collectively such variability can impact the whole simulation and profiles collected can be used to verify the prediction models built as part of Task 4.5.
- Energy optimization: By accessing the performance profiles of the whole multiscale simulation it will be possible to identify phases in which sub-simulations are completing too early. These sub-simulations may be candidates for techniques such as frequency scaling, which can deliver the same scientific results in the same time but with a lower energy footprint.
- Optimization of complex workflows: Detailed performance analysis allowing for system level, environmental and source-level bottlenecks to be identified can be done using Allinea tools. The extensions to the tools provide the ability to view the overall performance of a multi-scale simulation as well as source-level detail. This allows for identification – right down to the source code – of changes that can optimize complex workflows executing on possibly geographically distributed computers, where the changes may not have a large impact when considered within a single scale model.

2.2.3 Performance prediction models

We will develop performance prediction models for our multiscale applications for two reasons. First, we need performance information on the level of the single scale models and the conduits to provide sufficient information to the multiscale computing patterns code generators (2.2.2) such that they can interact with the middleware to define an optimal execution scenario. Next, we need performance

prediction models to be able to project the performance to yet non-existing exascale machines. Given our models, validated against runs on the EEE, we will investigate performance on emerging exascale architectures.

This is achieved in three steps. First, a mathematical model of the runtime measures will be made and validated. Secondly, a simulation of the runtime and a prediction of its variability will be performed, taking into account the scheduler, queuing times, system failures, energy use and system heterogeneity. Measures may include the amount of CPU hours used or data generated, but also the amount of energy consumed. Finally, the effect of the patterns on actual simulations will be measured throughout the project, creating a view of the cumulative effect of implementing different parts of the computing patterns and progressing application development as well as analysing their behaviour on emerging exascale machines.

2.3 ComPat applications

ComPat is a science driven project. The urgent need to push the science forward, and stay world leading in simulation driven science and engineering is our major motivation. The challenges in each of the domains represented in ComPat are of enormous intellectual, societal, economic and industrial concern. The first is in nuclear fusion, where our ability to understand interactions between turbulence at very small scales and the large scale plasma behaviour holds the key to control its magnetic confinement in order to produce clean and carbon free energy for the indefinite future. The second, from Astrophysics, aims to understand the formation processes of stars in their clustered environment, as well as the origin and propagation of structure in the stellar disk of the Milky Way Galaxy. A third has the highly ambitious aim of the predicting the materials' properties of macroscopic samples of matter based on the specification of the atoms and molecules comprising it. Our final challenges are in biomedicine: we need to obtain deeper understanding of pathophysiology of vascular disease, to provide personalised models of the vasculature in near to or real time for the purpose of supporting of clinical decision-making, and to rapidly and accurately calculate the binding free energies in drug discovery and personalized medicine.

Each of these applications will use the integrated nature of the ComPat system architecture to support and extend their complicated multiscale scenarios. In this section, we show, for selected examples, how the multiscale scenarios envisaged will be made possible by the ComPat architecture.

In the domain of biomedicine, we have recently calculated the free energy of binding of an unprecedented number of ligands to proteins by utilizing the whole of the SuperMUC machine at LRZ (250,000 cores in total). To achieve this feat, we required a highly specific set of scripts and commands, combined with human interaction, to ensure successful completion of a workflow composed of a large ensemble of interdependent replica simulations. With the ComPat system architecture, the submission of ensembles can be handled and abstracted by Pilot Jobs, the simulations can be monitored and analyzed

by QCG monitoring and Allinea profiling respectively, and the data can be automatically transferred back to our host institution using the Data Transfer software. The ComPat architecture will therefore make the pre-execution and execution of our simulations easier and automated, as well as more robust, reusable and with shorter turnaround times. It will also allow us to straightforwardly extend this methodology to a range of new use cases and create more complicated workflows (for example, re-simulation and modification of promising drug candidates, which will require multiscale coupling to quantum simulations for parameterization). In the future, we envisage using distributed HPC resources for this application when we are not able to access a large amount of resources on a single machine; the ComPat software architecture will make this possible through its middleware services, such as the QCG-Broker and the Jobs and Advanced Reservation Manager. Using performance prediction models and the Replica Computing code generator, the QCG tools will be able to schedule and execute our free energy of binding simulations in the most efficient way (including energy efficiency), with the minimum of human interaction. Due the generic nature of the computing patterns, this schema will be reusable for other Replica Computing applications as well.

Similarly, the ComPat architecture will allow advanced multiscale simulations scenarios to become possible for the other applications. In the fusion domain, the cornerstone component in both applications is MUSCLE, which couples together the singles scale models. In the HMM fusion application, advance reservation and energy aware components (ECOL, QCG scheduler), in combination with the HMM manager, will improve the efficiency of the application on the targeted computing systems. For the Aneurysm Flow Dynamics application, which currently uses the FabSim toolkit for job submission and curation of the single-scale simulation code (HemeLB), the QCG brokering system and the advanced reservation system, combined with MUSCLE, will allow the coupling required for the Extreme Scaling computing pattern. The ComPat architecture will allow this to be integrated with the workflows we currently use with the FabSim tool.

For each application, tools in the ComPat architecture are required to make their multiscale simulations viable. In Table 3 we identify the components of the architecture that will be used by each application. As we can see, many of the components will be used by multiple applications, indicating the generic nature of the architecture.

More detailed information of the application multiscale simulation scenarios, their use and integration of the ComPat architecture will be presented in Deliverable 3.1. In this rest of this section, we describe the libraries and toolkits required by our single-scale applications.

		Application and pattern									
		Nanomaterials	Biomedicine					Fusion		Astrophysis	Astrophysics
		"on-the-fly" coarse-graining (HMM) (RC)	Aneurysm flow dynamics (ES) (HMM)	In-stent restenosis (ES)(RC)	Binding affinities (RC)	RBC and platelet transport (ES)	Blood rheology (HMM)	Global turbulence simulation (ES)	Flux-tube chain (HMM)	Milky-Way Galaxy simulation (ES) (HMM)	Star cluster formation simulation (ES) (HMM)
ComPat high-level tools	QCG client tools	√	√	√	√	√	√	√	√		
	FabSim	√	√		√						
	Monitoring					√	√	√	√	√	√
	HPMC Pattern Code Generators	√	√		√		√	(√)	(√)		√
	Debuggers						√	√	√		
	Profilers	√	√	√	√	√	√	√	√	√	√
	Performance Prediction Models	√	√	√	√	√	√	(√)	√	√	√
Application libraries and parallel programming toolkits	ADIOS (Parallel I/O)					√			√	√	√
	MPI	√	√	√	√	√	√	√	√	√	√
	CPU Parallel Library	√	√							√	√

	GPGPU Parallel Library	√								√	√
	ECOL		√			√			(√)		
Patterns' libraries	MUSCLE	√	(√)	√		√	√	√	√		
	MPWide	√	√	√	√	√	√		(√)	√	√
	AMUSE									√	√
Patterns' services	Pilot jobs				√						
	HMM Manager	√					√		(√)		√
	On-the-fly database	√		√					(√)		
Middleware Services	Energy-aware Scheduler /Resource coordinator	√	√	√	√	√	√		√		√
	Jobs and advance Reservation Manager	√	√	√	√	√	√	√	√		√
	Workflow Manager	√		√	√		√	(√)	(√)		
	Data Transfer (grid FTP/ Globus)	√	√	√	√	√	√	(√)	√	√	
	ECOS	√	√	√		√	√		√		
	MTO		(√)	√		√	√		√		

Table 3 The required components of the ComPat architecture for each application. The ticks in brackets indicate that the use of the component depends on the development of the application

2.3.1 Application libraries and parallel programming toolkits

The basic role of application libraries and parallel programming toolkits in the architecture of the ComPat system is to provide necessary parallelization support for single-scale application kernels. To this end, a set of popular parallelization technics, natively supported by the ComPat middleware and Allinea tools, will be used. Moreover, our expectation is that computationally intensive ComPat kernels will be successfully integrated with the new ECOL library for optimization of energy consumption.

2.3.1.1 MPI, OpenMP, CUDA, OpenCL

MPI is the de-facto standard for computing in the distributed memory model, while OpenMP threads play the same role for the shared memory model. These two approaches are fundamental also for ComPat. However, many new codes make use of more recent computer architectures, such as GPGPU or ARM, that need to be handled with different technologies. The scientific codes are often implemented in various hybrid configurations, so an MPI application is frequently supported by OpenMP or CUDA. This leads to conclusion that the ComPat system, particularly the middleware and EEE, has to support a broad range of core libraries for parallelization of single-scale kernels and it has to be flexible and open enough to handle state-of-the-art and emerging approaches. In the current phase of the project we already provide support for MPI, OpenMP and CUDA. It is also possible to utilize any hybrid combinations of them.

2.3.1.2 Parallel I/O libraries

The amount of data generated by the applications, especially in the Extreme Scaling computing pattern, can be enormous. For big simulations it is not enough to provide sufficient storage capacity, but the application has to have guaranteed and efficient access to the data. For a massively parallel application writing checkpoint files, this can be a challenge. In addition, this is needed for the Extreme Scaling patterns for fault tolerance. A step towards tackling this is using a specialized parallel I/O library to increase the bandwidth and the general throughput of I/O devices. In this way, portions of data are spread over processes to efficiently write the data to standard file formats. In the ComPat project parallel I/O libraries will be used directly by applications to mitigate the input/output (I/O) bottleneck.

2.3.1.3 Energy Consumption Optimization Library (ECOL)

To make the patterns energy efficient, it is important to control at runtime the properties of various functional units of heterogeneous computing resources, e.g., dynamically turn on and off individual cores or change the clock frequency of processing units on-demand. At the same time, the completion time of the overall application should not increase above the accepted threshold. As the state-of-the-art has significantly changed since the time when the energy-efficiency subsystem of ComPat was proposed in the DoW, we are going to perform a set of experiments to determine the most rational way of dealing

with the energy consumption optimization. To this end we are going to compare capabilities of two the most popular Linux kernel drivers for scaling CPU frequency, namely the old one – ACPI cpufreq – and the new one, becoming a default for modern Intel CPUs – intel-pstate. Both of them differ in the range of the offered governors' scaling policies. The most significant difference is lack of the userspace governor in the current implementation of the intel-pstate driver, what results in lack of possibility to set manually the frequency. This capability is crucial for the initially proposed approach and lack of it results in necessity to re-design the concept. Therefore, in our research we are going to investigate if some improvements in power saving can be achieved not only by direct adaptation of the frequency but also by changing dynamically the configuration of the selected governor. The cost-benefit analysis in comparison to the standard native linux mechanisms will be crucial for the decision how the energy aspects will be addressed in ComPat.

We assume that incorporating energy awareness at the application level will be through the development and deployment of a new Energy Consumption Optimization Library (ECOL). ECOL will provide a well-defined API that will allow to dynamically modify properties of resources on which sub-models (components of application) are running. It will be for use by an application itself or any service controlling the execution of an application to downsize the overall energy consumption. The final decision whether ECOL will work in ComPat on the level of a single scale model, a coupled multiscale application, or both will be based on the results of our experiments. In order to support energy-efficient execution of ComPat applications, ECOL will provide remote access to advanced dynamic power management policies at the HPC hardware level controlled and offered by Energy Consumption Optimisation Service (ECOS). Using the ECOL semantic it will be possible to mark logical phases of the application's sub-model that should be reflected in specific resources' configurations (like governor and its configuration). Thus, depending on the CPU- and non-CPU-bound phases of the application, the characteristics of the resources (e.g., CPU frequency/voltage scaling or parameters of current scaling policy) could be dynamically changed only if it is not restricted by local computing resource policies. In turn, the optimization of the energy consumption of the whole application can be realised by adaptation and synchronization of the performances of its sub-models. Such a solution should bring a significant reduction of power consumption with negligible degradation of the application performance. In order to support all ComPat applications, ECOL will provide bindings for all languages in which the ComPat single scale models are or will be implemented.

2.4 ComPat patterns

We define multiscale computing patterns as high-level call sequences that exploit the functional decomposition of multiscale models in terms of single scale models. We have identified three computing patterns that, based on our previous experience with multiscale modelling, we believe to be most relevant for high performance multiscale computing, namely**:**

- Extreme Scaling,
- Heterogeneous Multiscale Computing, and
- Replica Computing.

A major design decision for ComPat, is the realisation that the Multiscale Computing Patterns will be expressed on the level of the task graph, generated from xMML, which is then taken together with execution recipes specific for a pattern and performance models or data for the single scale models and scale bridging algorithms, to determine the actual execution, which will then be specified in configuration files for the ComPat middleware. The formulation will provide enough information to determine runtime aspects, such as number of cores needed, average amount of time used or energy consumed. Equipped with this formal specification, the high-level tools will be able to provide estimates for the runtime behaviour of a model on different resources and produce runtime configuration files.

2.4.1 Patterns' libraries

In the following subsections we present libraries that will be implemented or used within the patterns. These libraries will be accessible by both the users and the middleware.

2.4.1.1 MUSCLE

The Multiscale Coupling Library and Environment (MUSCLE) is a formal specification and framework for implementation and execution of multiscale models on distributed resources, including HPC. In the project we consider usage of two implementations of the MUSCLE specification, namely MUSCLE2 and MUSCLE-HPC, which is a recent development from the University of Geneva (please note that they were partners in the MAPPER project, and that the PI of the team behind MUSCLE-HPC, Prof. B. Chopard, is member of the Scientific Advisory Board of ComPat).

MUSCLE2 [8,9] is a component-based framework. It recognizes the temporal scale of single scale models and keeps simulation time between submodels in sync when they communicate. From a runtime perspective, the submodels may be written in Java, C, C++, Fortran, Scala, Python, or Matlab, in combination with any parallelisation mechanism. Execution on distributed resources is made possible by a dedicated asynchronous and multi-channel message forwarder.

Submodels in MUSCLE are computed independently so in the Replica Computing pattern they can be independently analysed. For the Extreme Scaling pattern, MUSCLE has a high-performance setup and acknowledges MPI processes. It will be extended to allow parallel connections for every submodel, so each MPI process or each node can set up its own connection. This way, data is not unnecessarily moved within the submodel. For optimally using the Heterogeneous Multiscale Computing pattern, MUSCLE should be able to dynamically add new submodels to a simulation, so that the micro-scale simulations can be started and restarted as needed. These added features fit into the overall architecture of MUSCLE and will be realised early in the project.

MUSCLE-HPC is a low latency, high bandwidth version of MUSCLE optimized for running multiscale simulations on a single cluster. The Muscle-HPC is designed compliantly to the MMSF approach with the advantage of using the MPI communication interfaces to couple the codes. It provides an MPI native interface for C/C++ and the concept can also be applied to other languages which implements the MPI-2.2 features like in Fortran and Python (the current API is developed and tested using C++ language). MUSCLE-HPC provides a comparable performance to the native MPI computation time, while keeping the high level design principle of MMSF.

2.4.1.2 MPWide

MPWide is a communication library for distributed message passing, optimised for performance, used for both parallelizing simulations across multiple distributed computing resources and for ensuring highly responsive coupling between single scale models in a multiscale simulation [10,11]. MPWide is embedded into MUSCLE and was directly used to facilitate very fast coupling in a multiscale blood flow application [12]. MPWide supports models written in C, C++, Python or FORTRAN.
We will adapt MPWide in four ways. First, we are going to implement automatic performance tuning of communication paths, improving performance without decrease of usability. Second, we will implement mechanisms to explicitly detect and classify connection errors, so that MPWide can use its runtime connection/disconnection routines to facilitate fault-tolerant operation. Third, MPWide will be extended to allow use of multiple network routes for its communications, both to boost performance and to introduce fault-tolerant communications. And fourth, several new non-blocking and asymmetric message-exchange mechanisms will be implemented to increase the utility of MPWide for new applications.

2.4.1.3 AMUSE

The Astrophysical MUltipurpose Software Environment (AMUSE) provides a homogeneous interface to a wide variety of packages, which enables the research of astrophysical phenomena where complex interactions occur between different physical domains [13,14]. The natural hierarchy makes distributed computing relatively straightforward, but load balancing and high performance is complicated due to the wide variety of compute requirements and the diversity of algorithms. Some algorithms work excellently on parallel GPU architectures, whereas others require more traditional architectures. For astronomical research generally a wide variety of codes are coupled via the AMUSE framework, and the specific runtime requirements of each of these codes makes for an enormous challenge. Optimizations are required for networking, runtime behaviour, data I/O, the generation of initial conditions and raw flop rates for production. The multiscale computing patterns proposed here will enable a smooth porting of the framework to exascale infrastructures.

2.4.2 Patterns' services

2.4.2.1 Pilot jobs

Energy-aware allocating of large-scale resources will be done through the QCG middleware stack. However, running a large and *a priori* unknown number of relatively small jobs in a queuing system requires the use of a specialized solution to ensure proper level of utilization of resources and to mitigate the negative impact of cumulative queuing time on the overall performance of the multi-tasks simulation. The well-known and wildly used solution is to benefit from the concept of a pilot job, in which all the tasks are executed inside a single meta-job allocation. Systems implementing the pilot job concept provide flexible application-level resource management capabilities to efficiently utilize allocated resources. The most suitable implementation of the pilot job approach will be chosen based on a detailed analysis, taking into account not only the available functionality but also the ease of integration with the whole ComPat stack. One of the candidates is the RADICAL Cybertools Pilot Job implementation, which provides the ability to concurrently start multiple instances of single scale models by packing them into only one job slot on an HPC resource.

2.4.2.2 HMM manager

A central component in Heterogeneous Multiscale Computing is the HMM manager, which coordinates the data flow from the fine-scale models, which run on-demand, and the coarse-scale model. The HMM manager also assesses the possibility to forego on-demand execution of a fine-scale model, and pass back values based on interpolations from existing results instead. To implement the HMM manager we will rely on a non-centralized database solution (on-the-fly-database), to be wrapped by MUSCLE component, in combination with a parallelised I/O library. The manager thus becomes a new component in MUSCLE, on the same level as 'mappers', 'conduits', or 'filters'. The main difference is that the manager should be able to keep its state (that is, store it to disk) for subsequent runs (thus avoiding recomputing the whole content of the database over and over again). Note that user communities could decide to set up a central data base solution (maybe even in the public domain) to share computed HMM results, and that the HMM manager actually reads from such central database. This is however beyond the scope of ComPat. The manager will be designed in such a way that such further innovations are possible.

2.4.2.3 On-the-fly database

We will construct an on-the-fly database to limit the number of microscale simulations required [15, 16]. This database serves to store previously computed data and, where desirable, interpolate between already computed values to provide input to the macroscale model. This is feasible because the amount of data passed up to the macroscale simulation is usually not large, perhaps a few floating-point numbers representing quantities of interest (such as the viscosity for fluid problems). The on-the-fly database will

be managed by the HMM manager, and can also play a role in the Replica Computing pattern (depending on the exact usage mode of the RC pattern).

2.5 ComPat middleware services

In order to simplify and automatize scientific simulations, the access to distributed HPC resources is typically made via a middleware systems. There is no exception in ComPat, where pattern-enhanced multiscale simulations, as shown in Figure 3, are submitted, controlled and managed by the QCG (former QosCosGrid) Middleware. The role of QCG and other middleware services is a subject of the next few sections. Please note that the figure - for completeness - includes black-boxes called patterns' services. The functions of components embedded in these black-boxes is out of scope of this section and it is discussed in section 2.4.2 along with a detailed description of particular patterns.

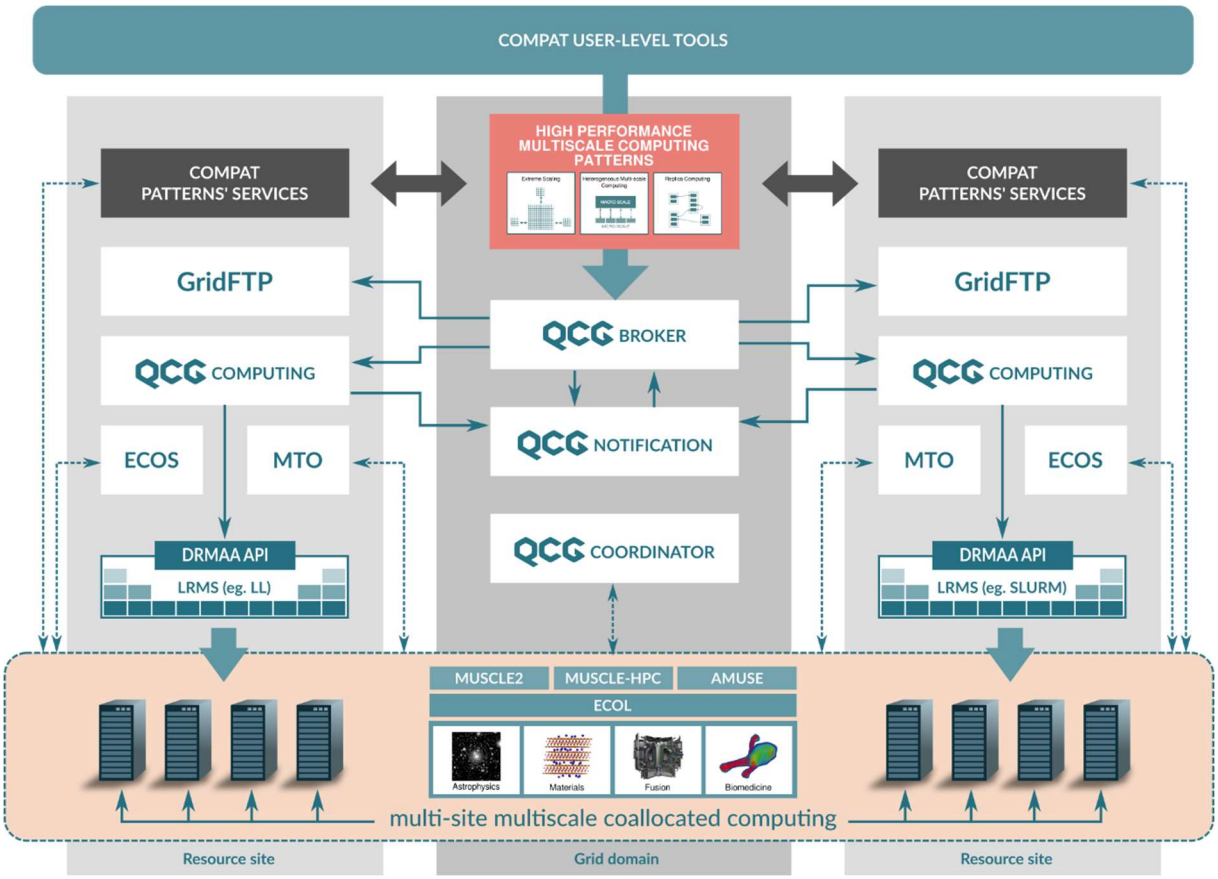

Figure 3. The architecture of ComPat middleware layer

2.5.1 QCG-Middleware

The QCG [17,18,19] middleware can virtually weld computing resources from different administrative domains into a single powerful supercomputing resource. This capability is essential for the ComPat applications to scale up towards exascale performance already now, while there are only petascale systems available. The middleware has been developed in the EU project QosCosGrid and later adapted and extended to support requirements of multiscale codes in the MAPPER project. Presently QCG

delivers a ready-to-use stack of validated efficient middleware software components that can manage extremely parallel as well as multiscale simulations, notably MUSCLE-based ones. Our analysis and initial tests confirm that the functionalities offered by the midleware provides solid foundations for innovative ComPat scenarios. The middleware will, of course, need to be improved and extended to address heterogeneous architectures and specific needs of the ComPat project. Especially, the energy efficiency aspect of multi-scale calculations is a completely new functional area that has to be thoroughly analysed in order to integrate with the QCG stack. The extensions to the middleware services will be done in close cooperation with other Work Packages.

2.5.1.1 Energy-aware scheduler and resource co-allocator (QCG-Broker)

The Energy-aware Scheduler will be designed and implemented as a new module to the QCG-Broker service. The module will be responsible for taking multi-criterion scheduling decisions for assigning jobs to resources with the node granularity, based on its knowledge of application profile (including energy characteristics) and topology of underlying resources. The middleware will schedule and co-allocate jobs to minimize the cost of infrastructure operation trying to guarantee the requested level of Quality of Service or, at least, minimize its violations. To optimize energy consumption, jobs submitted by the middleware may also be run at lower than nominal CPU frequency.

Before its use in production, the new scheduling/brokering policies, as well as resource management ones, will be verified and tested within the specialised simulation environment. To this end, the PSNC's DCworms simulator will be used. This tool allows modelling and simulation of computing infrastructures to estimate their performance and energy consumption for diverse workloads and management policies [20]. It allows modelling of computing patterns with various granularities. The presence of detailed resource usage information, current resource energy state description and functional energy management interface enable relatively easy implementation of novel energy-aware scheduling algorithms where resource energy consumption becomes an additional criterion in the scheduling process. Based on performance models for the multiscale computing patterns, DCworms will be used to test and evaluate various scheduling and resource management strategies for exascale multiscale applications.

2.5.1.2 Job and advance reservation manager (QCG-Computing)

QCG offers capabilities for managing jobs and advanced reservations on a single resource through the QCG-Computing service. The component may be used standalone to provide access to a single resource, however in complex scenarios, such as ComPat multi-scale simulations, it is heavily exploited by QCG-Broker to manage execution of co-allocated multi-site jobs. QCG-Computing proved to achieve high performance and can be considered as one of the most reliable and efficient tools that provide remote access to queuing systems and their capabilities including advance reservation [21]. In scope of the

project the QCG-Computing service will be extended to provide to QCG-Broker information needed for energy-aware scheduling and brokering of jobs as well as to support jobs requiring many heterogeneous resource allocations.

2.5.1.3 Workflow manager (QCG-Broker)

QCG-Broker is able to deal with complex applications defined as a set of tasks with precedence relationships, i.e. workflows. What differentiates QCG from other middleware services supporting workflows is that every single task can be connected not only with input or output files, but it may be also trigged by predefined conditional rules or any status of one or more jobs or tasks. In the project QCG Workflow Manager will be either extended to support specific needs of patterns or new API interfaces will be exposed to allow implementation of specific patterns' business logic on the pattern or application side. It is also foreseen to develop necessary extensions that will provide fault tolerance on various levels of the multi-scale application execution.

2.5.1.4 QCG-Notification

The QCGNotification service role is twofold. On one hand it is actively used for asynchronous communication between QCG middleware services. On the other hand, it provides users with the basic application's progress tracking functionality by means of e-mail and XMPP notification messages. In ComPat the QCG-Notification service together with the QCG-Monitoring portal assist users in estimating the correctness and progress of remote calculations, thus helping them to reduce waiting time and improve the overall performance. These two features were regarded as essential by ComPat application groups.

2.5.1.5 QCG-Coordinator

The multi-resource execution of jobs requires special treatment. Among others it is not straightforward to discover and to connect application's modules distributed over many resources to enable simultaneous start of a simulation. In order to deal with this problem, the QCG middleware utilise a lightweight QCG-Coordinator service that gathers knowledge about running tasks.

2.5.2 Data transfer (GridFTP / Globus Transfer)

Data transfers of huge amounts of data, either between the distributed resources used by the multiscale applications, or to and from the home institutions of the domain scientists, takes longer than it has in the past. Very often, it is required that the system by itself will cope with transfer interruptions and data corruption during the transfer. In ComPat we assumed that primary data transfer mechanism will be GridFTP [22], which is a mature and widely available solution providing a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. In case of insufficient capabilities of the plain GridFTP protocol, we will employ Globus Transfer (formerly known as Globus

Online) that assures data integrity, and can automatically restart interrupted transfers. Globus Transfer is a cloud-based coordination service that works with standard GridFTP endpoints (like those used in QCG), thus it can be easily integrated with QCG using the Globus Transfer's highly reliable fire-and-forget approach.

2.5.3 Energy Consumption Optimization Service (ECOS)

The role of Energy Consumption Optimization Service is the management of energy parameters of single or many resources involved in computations of a distributed multiscale application. Based on dynamic information provided by means of the Energy Consumption Optimization Library, the service will be responsible for setting up the best possible configuration of all resources in terms of energy usage. The main scenario assumes that changes of properties of resources are requested by an application and correspond to its phases, but one can imagine scenarios in which ECOS is integrated with the energy-aware scheduler or with any pattern manager.

2.5.4 Multisite Transport Overlay (MTO)

Multisite Transport Overlay is a successor of the MUSCLE Transport Overlay service used in the MAPPER project. It is a daemon for forwarding traffic across administrative domains. This service is often required when application kernels, in particular those that are MUSCLE-based, must communicate between several sites with firewall restrictions or Network Address Translation (NAT).

2.6 Experimental Execution Environment

Within the project, we realise an Experimental Execution Environment, which is continuously available to the project, with a fully functional technology stack as described above. The components of the ComPat stack described in this deliverable are being deployed and made available for end-users in Work Package 6 - see Deliverable 6.1 for details. In its current state, the EEE already provides all the middleware services described in Section 2.5 of this document. The environment will allow for efficient testing, benchmarking, and even small production runs of ComPat's applications. It is fully monitored and quality assurance processes are in place. With time, the EEE will adapt to the needs of its users, integrating new components.

3 Conclusions

The architecture of the ComPat system presented in this document addresses the needs identified so far in the project. We believe that the proposed multi-layered design and division in fast-track and deep-track components not only allows to develop a fully functional system capable of running highly demanding applications on extremely large infrastructures, but also facilitates and systematizes the work in the project. The presented architecture will be implemented, deployed and verified in the next phases of the project in accordance with the Operational Model accepted and described in deliverable D6.1.

4 Annexes

Annex A - Middleware Capabilities Questionnaire

1. brokering capabilities: matching applications or their parts to resources to guarantee SLA and to ensure requested trade-off between the application performance and the cost of resource energy consumption
2. advance reservation of computing resources
3. advance reservation of network bandwidth (will be implemented if really needed)
4. co-allocation of resources: advance reservation of many (potentially heterogeneous) resources to synchronize simultaneous processing of many tasks constituting a single application on these resources
5. [energy] efficient data transfer – staging in and out files and directories in an [energy]efficient way
6. fault tolerance – resubmission of an application module (auto-restarting) when this functionality is not supported directly by the LRMS (queueing system) or the provided functionality is not sufficient.
7. checkpointing and resuming of an application or its part (we can imagine that the middleware can force an application to checkpoint itself, for example to resume calculations on another resource. it will require discussion what triggers such action)
8. application driven dynamic adaptation of performance and energy consumption characteristic of the resources via the ECOL library and ECOS services to adapt the characteristic to requirements of current phase of the application
9. support for the execution of an application / module on hybrid CPU/GPGPU systems
10. workflow management – currently the DAG model is supported and the execution of a child task can be triggered by any combination of status changes of a parent tasks. (we need to know if more advanced constructions like conditions or loops based on tasks' statuses and application's results are needed - so far they are not supported)
11. rescheduling – (automatic) resubmission/migration of a task or its part to another resource to improve the overall system or application metric (needs clarification, for example what should trigger such action)
12. various types of available middleware clients: commandline, desktop, portal. The commandline is prefered for advanced application scenarios especially in the first phase of the project, but the other choices may be considered for the future.
13. JAVA / Python API (to use the middleware functions by external tools or from the inside of the application)

14. Visual (portal based) monitoring of progress of long running applications (including energy consumption)
15. in-situ visualization – visualization of the application results in parallel to the computations (paraview), possibility of steering of an application execution by altering its parameters in the run time
16. parameter sweep – support for multi-dimensional parameter sweep jobs. Due to performance reasons a number of sub-tasks is limited and for running hundreds of tasks we strongly recommend application side solutions, e.g. pilot jobs
17. load balancing – mapping/assigning jobs to resources in a way that balance the overall load on the system (cluster level mechanism).

Annex B – Resource Information Questionaire

	A	B
1	**Institution:**	
2	**Resource name:**	
3	**Hostname and ports:**	
4	**Hardware Info:(URL to such information or a full description in text)**	
5	**Batch Scheduler:(Including version)**	
6	**Availability of Advance Reservation:(Please include also how many days in advanced the request has to be made and any other special restrictions/requirements)**	
7	**Availability to modify energy/performance characteristics:(Please include any restrictions/requirements)**	
8	**Local Policies with installation of additional libraries, applications and services:**	
9	**Data Storage on Resource:(Please include also the name of such variables on the local system, e.g. $PROJECT, $WORK, $SCRATCH, and their quota restrictions)**	
10	**Offered Services for Data Transfer:**	
11	**Local security and Firewall Policies:**	
12	**Local Authentication and Authorisation Policies:**	
13	**Local Accounting Policies:**	
14	**Local Monitoring Tools and Services:**	
15	**Any Other Crucial Local Policies or Information:**	

5 References

1 Derek Groen, Agastya Bhati, James Suter, James Hetherington, Stefan Zasada, Peter V. Coveney, FabSim: Facilitating computational research through automation on large-scale and distributed e-infrastructures, Computer Physics Communications (doi:10.1016/j.cpc.2016.05.020)

2 Allinea DDT, the parallel debugger [Online], Availiable: [Accessed 15 June 2016]

3 Allinea DDT Flyer [Online], Available: , [Accessed: 15 June 2015]

4 Allinea MAP, the parallel profiler [Online], Available: , [Accessed: 15 June 2016]

5 Allinea MAP Flyer [Online], Available: , [Accessed: 15 June 2016]

6 Allinea Performance Reports, characterize and understand the performance of HPC application runs [Online], Available: , [Accessed: 15 June 2016]

7 Allinea Performance Reports Flyer [Online], Available: , [Accessed: 15 June 2016]

8 Borgdorff, J., M. Mamonski, B. Bosak, D. Groen, M.B. Belgacem, K. Kurowski, and A.G. Hoekstra, Multiscale Computing with the Multiscale Modeling Library and Runtime Environment. Procedia Computer Science, 2013. 18(0): p. 1097-1105.

9 Borgdorff, J., M. Mamonski, B. Bosak, K. Kurowski, M. Ben Belgacem, B. Chopard, D. Groen, P.V. Coveney, and A.G. Hoekstra, Distributed multiscale computing with MUSCLE 2, the Multiscale Coupling Library and Environment. Journal of Computational Science, 2014. 5: p. 719-731.

10 Groen, D., S. Rieder, P. Grosso, C.d. Laat, and S.F.P. Zwart, A lightweight communication library for distributed computing. Comput. Sci. Disc., 2010. 3: p. 015002.

11 Groen, D., S. Rieder, and S. Portegies Zwart, MPWide: a light-weight library for efficient message passing over wide area networks. Journal of Open Research Software, 2013. 1: p. e9.

12 Groen, D., J. Borgdorff, C. Bona-Casas, J. Hetherington, R.W. Nash, S.J. Zasada, I. Saverchenko, M. Mamonski, K. Kurowski, M.O. Bernabeu, A.G. Hoekstra, and P.V. Coveney, Flexible composition and execution of high performance, high fidelity multiscale biomedical simulations. Interface Focus, 2013. 3(2).

13 Portegies Zwart, S., E.v. Elteren, and I. Pelupessy, Multi-physics simulations using a hierarchical interchangeable software interface. Computer Physics Communications, 2013. 183: p. 456-468.

14 Portegies Zwart, S.F., S.L.W. McMillan, A. van Elteren, F.I. Pelupessy, and N. de Vries, Multi-physics simulations using a hierarchical interchangeable software interface. Computer Physics Communications, 2013. 184(3): p. 456-468.

15 Lorenz, E., Multi-scale simulations with complex automata: in-stent restenosis and suspension flow. 2011,PhD thesis, University of Amsterdam.

16 E, W. and B. Engquist, The Heterogeneous Multiscale Methods. COMM. MATH. SCI., 2003. 1(1): p. 87-132.

17 QCG - Quality in Cloud and Grid. Available from: [Accessed: 15 June 2016]

18 Bosak, B., P. Kopta, K. Kurowski, T. Piontek, and M. Mamoński, New QosCosGrid Middleware Capabilities and Its Integration with European e-Infrastructure, in eScience on Distributed Computing Infrastructure. 2014, Springer. p. 34-53.

19 Bosak, B., J. Komasa, P. Kopta, K. Kurowski, M. Mamoński, and T. Piontek, New capabilities in QosCosGrid middleware for advanced job management, advance reservation and co-allocation of computing resources – quantum chemistry application use case, in Building a National Distributed e-Infrastructure–PL-Grid,. 2012, Springer. p. 40-55.

20 Kurowski, K. and e. al., DCworms–A tool for simulation of energy efficiency in distributed computing infrastructures. Simulation Modelling Practice and Theory, 2013. 39: p. 135-151.

21 Radecki, M., T. Szymocha, T. Piontek, B. Bosak, M. Mamoński, P. Wolniewicz, K. Benedyczak, and R. Kluszczyński, Reservations for Compute Resources in Federated e-Infrastructure, in eScience on Distributed Computing Infrastructure. 2014, Springer. p. 80-93.

22 Globus GridFTP [online], Available: http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/ [Accessed 15 June 2016]