



## Initial Agent-Based Model Work Package 1 Tasks 1.1-1.3 Deliverable 1.2

### Authors

Artikis, Alex – NCSR  
Atsidakou, Alexia – NCSR  
Michelioudakis, Evangelos – NCSR  
Montagud, Arnau – BSC  
Monti, Michele – CRG  
Ponce de León, Miguel – BSC  
Pradas, Gerard – BSC  
Saxena, Gaurav – BSC  
Tartaglia, Gian Gaetano – CRG/ITT  
Valencia, Alfonso – BSC  
Vicente, David – BSC

 European Commission Horizon 2020 European Union Funding for Research & Innovation	Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3 Deliverable D1.2</b>	<b>Doc.nr.:</b> WP1 D1.2
			<b>Rev.:</b> 1.0
			<b>Date:</b> 30/04/2020
			<b>Class.:</b> Public



## Distribution list:

Group:	Others:
WP Leader: BSC Task Leader: BSC	Internal Reviewer Partner: NCSR Demokritos (NCSR) INFORE Management Team INFORE Project Officer

## Document history:

Revision	Date	Section	Page	Modification
0.1	16/03/2020	All	1-3	Creation.
0.5	30/03/2020	All	1-25	All sections have been drafted.
0.8	09/04/2020	All	1-38	Version for internal review completed.
0.9	14/04/2020	All	1-39	Revision from NCSR performed.
1.0	28/04/2020	All	All	Revisions from NCSR incorporated.

## Approvals:

First Author:	Arnau Montagud (BSC)	Date:	09/04/2020
Internal Reviewer:	Evangelos Michelioudakis, Elias Alevizos (NCSR)	Date:	14/04/2020
Coordinator:	Antonios Deligiannakis (Athena)	Date:	29/04/2019

 Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3 Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



## Table of contents:

1	Executive Summary .....	4
2	Introduction.....	5
3	Agent-based modeling in Life Science use case .....	6
3.1	Use of agent-based modeling in biology	6
3.2	Definition of the agent-based modeling used in the present project.	7
4	Integration of the agent-based modeling in our multiscale Model.....	9
4.1	The agent-based component	9
4.2	The environment component	10
4.3	The signalling network module	12
4.4	The cell cycle module	13
5	Multiscale Model HPC Implementation.....	15
5.1	Design and parallelization	15
5.2	Experiments	19
5.3	Discussion	22
6	Extreme model exploration with optimization algorithms.....	25
6.1	Model exploration of simulation parameters	25
6.2	Use of different optimization metrics to find proper sets of parameters	27
6.3	Preliminary results	29
7	Conclusions and Future Work.....	31
8	References .....	32
9	Glossary .....	34

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h2>Deliverable D1.2</h2>	<b>Doc.nr.:</b>	WP1 D1.2
		<b>Rev.:</b>	1.0
		<b>Date:</b>	30/04/2020
		<b>Class.:</b>	Public



## 1 Executive Summary

This deliverable describes the agent-based model currently used in the Life Science use case of INFORE, the parameters used in its simulation and its HPC implementation. Additionally, we present preliminary results of a model exploration framework that is extremely useful for the study of models' parameters.

The ultimate goal of this use case is to provide a “*virtual laboratory*” for studying cancer growth and evolution by using multiscale models of tumors. Thus, by integrating a center-based agent model into a multiscale model (MSM) allows us to study different aspects crucial for the development and growth of tumors. Indeed, given the biophysical, biochemical, and biomechanical factors present in these problems, MSM can help identify the factors that drive a given treatment to be a success or a failure. This MSM consists of several components and modules that are hereby detailed and that operate at very different time scales: environment, cells, signaling pathways, and cell cycle behavior. Furthermore, to scale our simulations we need to parallelize our MSM. With that in mind we have started by parallelizing the environment component, which has the smallest time scale. This successful parallelization enables us to address the parallelization of the cells component, a task that is currently ongoing. These scaled-up simulations using the Barcelona Supercomputing Center (BSC) MareNostrum4 will incorporate forecasting techniques for various events of interest.

Lastly, we present a model exploration technique that allows us to define the structure and hierarchy of the model's parameters and to evaluate its sensibility to the parameters' perturbation. All these developments will facilitate the design of different set-ups that tally cancer tumor growth conditions with increased number of cells, altered microenvironmental physical properties, different cell types, as well as, study the interaction between cancer cells and the immune system.

 Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	<b>Doc.nr.:</b> WP1 D1.2
		<b>Rev.:</b> 1.0
		<b>Date:</b> 30/04/2020
		<b>Class.:</b> Public



## 2 Introduction

Modeling of signaling networks has proven its usefulness in a variety of health-related projects since it has allowed researchers to include vast amounts of data and study their behavior. Boolean modeling is particularly appropriate when the scientific question is qualitative, as is the case of many signaling studies, and it has been used for very different goals such as drug discoveries (Flobak *et al.*, 2015), high-throughput mutant analyses (Montagud *et al.*, 2017) and patient-specific outcomes (Béal *et al.*, 2019).

Multiscale models (MSM) are needed to model dynamics with very different timespans, as they can integrate cell signaling, cell-cell and cell-environment behavior, which take place at different time scales and use different mathematical approaches. For instance, multiscale modeling frameworks combining agent-based and Boolean models are useful as they can bridge from genes' activity to cells' phenotypes, to physical interactions among cells or cells with their environment. Agent-based models represent cells as single agents of a cell population and account for the interactions between cells, small diffusing molecules and the environment. These agents can move, grow, divide and stick to their neighboring cells and environment. Agent-based models have been successfully used to explore tumor spheroids and tumors boxed in ducts (Ghaffarizadeh *et al.*, 2018), to study defibrillation of a human heart in arrhythmia (Bernabeu *et al.*, 2010) and liver regeneration (Hoehme *et al.*, 2010). Boolean models, on the other hand, account for signaling pathways, cell cycle and cells' response to external signals that are integrated and can drive the cell to behave in a given manner: proliferate, migrate, divide, etc. Boolean models have been successfully used to predict mutants' effect on cancer phenotypes (Cohen *et al.*, 2015), cell sensitivity to drugs and the synergistic effects between pairs of drugs (Flobak *et al.*, 2015). Multiscale models are, thus, a promising genotype-to-phenotype mapping framework, which allows studying different kinds of variations and their effect on the cell's individual and collective behavior.

Models have been used to study genetic variations by evaluating the effect of all knock-outs mutants and over-expression of genes in cells' behavior (Montagud *et al.*, 2017). Likewise, environment variations effects on cells' behaviors have also been studied using models with interesting results. MSMs allow for the study of tumor growth and evolution, bridging the gap between different levels of description, and connecting events that occur at different scales (An, 2010). However, due to the uncertainties regarding the underlying biology, MSMs involve a high number of parameters that need tuning (Ozik *et al.*, 2018). Indeed, MSMs would benefit from the thorough exploration of the combination of these perturbations. To that end, high-performance computing (HPC) clusters, such as the Barcelona Supercomputing Center (BSC) MareNostrum 4 supercomputer, are ideal environments for intensive simulation of MSMs that produce extreme-scale data streams as outputs.

In the INFORE project, the Life Science use case is tasked to provide a “*virtual laboratory*” for studying cancer growth and evolution by using multiscale models of tumor systems (Trisilowati and Mallet, 2012). The goal of this use case is to facilitate the design, test, and optimization of cancer treatments based on combinations of different drugs and dosage strategies. In this deliverable we detail the agent-based model used in our MSM, its different modules and how they communicate among them. Also, we detail its internal parameters and their ranges. Additionally, we present the first version of our HPC implementation for this MSM framework. Lastly, we present and discuss some preliminary results on the model exploration performed on this MSM framework using a model of drug resistance.

 European Commission Horizon 2020 European Union Funding for Research & Innovation	Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
			Rev.:	1.0
			Date:	30/04/2020
			Class.:	Public

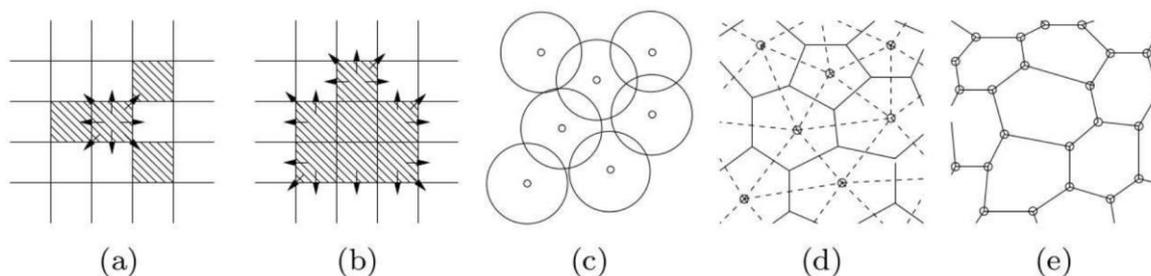
### 3 Agent-based modeling in Life Science use case

#### 3.1 Use of agent-based modeling in biology

Agent-based modeling (ABM) is a computational approach for modeling and simulating systems composed of autonomous agents, the environment, and the interaction between agents and the environment (Macal and North, 2010). ABM has been applied to model biological systems at different scales. For instance, the use of ABM to study population dynamics has a long tradition in ecology (Grimm *et al.*, 2006). Moreover, ABM has also been used to model biological systems at the cellular and subcellular scale (Anderson, 2005). One of the most relevant features of the ABM framework is that it allows simulating processes taking place at different time and space scales, e.g. diffusion processes and biochemical reactions, cell movement and mechanical interactions and population-level dynamics such as competition for resources.

In the field of cancer research, ABM has been used for modeling and simulating different properties of tumors. For instance, how tumor morphology can evolve driven by selective pressure from the microenvironment (Anderson *et al.*, 2006), the metabolic reprogramming in cancer cells (Shan *et al.*, 2018) or the stress response in growing tumor spheroids (Van Liedekerke *et al.*, 2019). Furthermore, multiscale ABM has been used to accelerate the discovery of immune-tumor interactions (Ozik *et al.*, 2019) and to optimize drug dosage and regime to improve treatment efficacy in solid tumors (Letort *et al.*, 2018).

From the modeling perspective, there are at least five different discrete or individual-based approaches that have been used to model a multicellular system (Osborne *et al.*, 2017). All these approaches treat individual cells as discrete entities but differ in the way the cells are represented. Also, these approaches have very different focuses, as they are meant to model different properties of the individual cells as well as their interactions. For example, the simplest model is the cellular automaton Figure 1a) where the domain is a lattice and each cell may be occupied by a cell-agent. Cellular automata have been used for studying the interaction between tumor cells and the extracellular matrix in 2D monolayers. Other approaches are depicted in Figure 1b-e (See Osborne *et al.*, 2017 for a complete review).



**Figure 1: Schematics of the most typical cell-based models considered in biology studies. (a) Cellular automaton. (b) Cellular Potts model. (c) Overlapping spheres model. (d) Voronoi tessellation model. (e) Vertex model. from (Osborne *et al.*, 2017).**

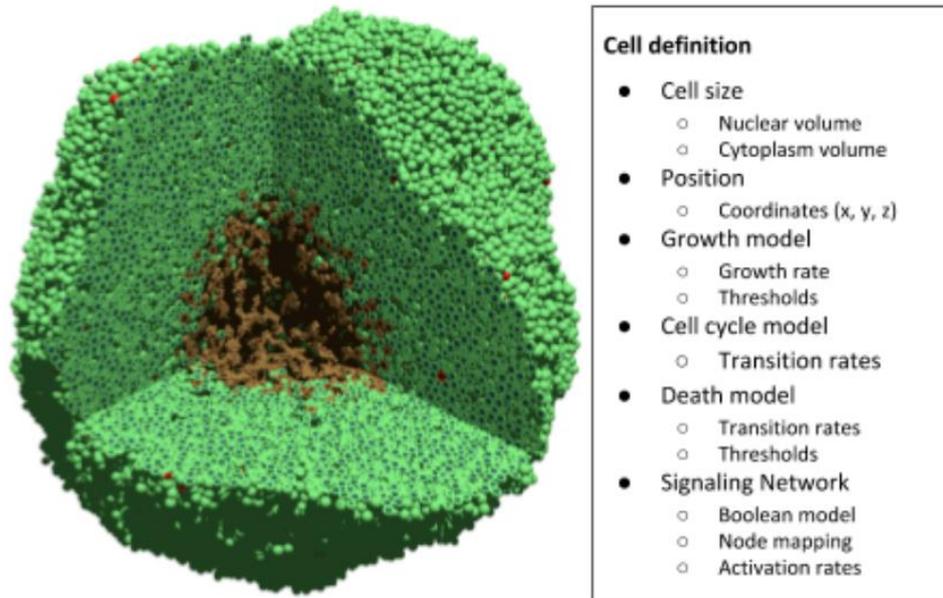
Furthermore, the different cell-based models depicted in Figure 1 are implemented and available as programming frameworks or standalone software packages. Some of the most widely used tools include Chaste (Mirams *et al.*, 2013), a general-purpose modeling package for multicellular systems in a lattice model; CompuCell3D (Swat *et al.*, 2012), a flexible scriptable modeling environment that implements a cellular Potts model (Graner and Glazier, 1992); CellSys (Hoehme and Drasdo, 2010), a modular software tool for efficient off-lattice simulation; and PhysiCell (Ghaffarizadeh *et al.*, 2018), an agent-based extensible framework for 3D multicellular simulations.

Herein we have focused on the overlapping spheres model (Figure 1c) as it can represent cells' movement, attachment and growth in a more implicit way than its more abstract alternatives. The overlapping spheres model was used to study the process of tumor cell growth in different arrangements or architectures (e.g. 2D monolayers, 3D spheroids) and treated with different drugs or combinations of drugs. Moreover, we have chosen the PhysiCell framework, together with its extension PhysiBoSS as the basis of our developments because of its OpenMP-based implementation and its extensibility. Nonetheless, OpenMP has a limited distribution and it's the reason behind our efforts towards an MPI-OpenMP implementation of PhysiCell in the section below.

<p>Project supported by the European Commission Contract no. 825070</p>	<p><b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b></p>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

## 3.2 Definition of the agent-based modeling used in the present project

As described in the previous section, the development of the agent-based model for simulating tumor growth and the responses to different drug regimens is implemented on the top of PhysiCell framework and is mostly based on its extension PhysiBoSS. As we detail in further sections, the PhysiCell framework manages the simulation of i) the environment, ii) the cell-agent mechanics, including movement and physical interaction and iii) the basic agent behavior by providing standard models for cell growth, division and death (Figure 2).



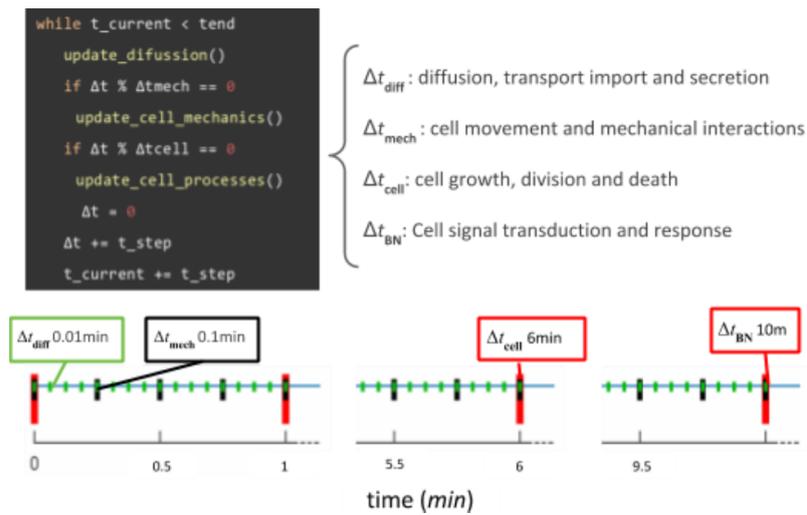
**Figure 2: Visual representation of a multiscale simulation of a population of cells. On the left side, a population of cells arranged in a 3D spheroid are depicted. Cell colors indicate cells exhibiting alternative phenotypes. For instance, the brown core at the center of the spheroid corresponds to necrotic cells that die because of the lack of nutrients. On the right side panel, the different attributes or properties of each individual cell agent are shown.**

Nonetheless, most of the behaviors of the cell agents are governed by complex rules that are not part of the basic agent-based mechanisms, as is the signaling transduction pathway, the intracellular machinery that integrate and process external and internal stimulus and transduce them into cell fate decision, e.g. start replication cycle or commit into apoptosis. In order to capture part of the intracellular processes and their dynamics, a model of cell signaling can be integrated within each individual cell-agent. This extension adds another layer to the multi-scale model allowing for a more detailed description of the signal transduction process which, for instance, can be used to simulate the effect of drugs in a more realistic way. This motivated the development of PhysiBoSS (Letort *et al.*, 2018), a framework which combines PhysiCell with MaBoSS<sup>1</sup>.

PhysiBoSS extends the functionalities of PhysiCell by allowing it to model and simulate the cell signaling network which processes inputs and dictates the cell fate (e.g. proliferate, commit apoptosis). Importantly, this extension adds a fourth time-scale in the model, the Boolean network time to the original three time-scales from PhysiCell: the diffusion, the mechanics and the cell division (Figure 3). At each time-scale, the engine queries the specific component and evaluates if it needs to update any of the component. The diffusion  $\Delta t_{diff}$ , typically 0.01 minutes, is the time where changes in the environmental entities are considered: their diffusion, reactions and transport. The mechanics  $\Delta t_{mech}$ , typically 0.1 minutes, is the time where changes in the cell physical behavior are considered: cell movement, cell-cell attachment, cell-environment attachment, etc.. The cell division  $\Delta t_{cell}$ , typically 6 minutes, is the time where PhysiCell queries if there is any change in the cell growth, cell division and the different death modes considered (Apoptosis and Necrosis). Finally, the Boolean network  $\Delta t_{BN}$ , typically 10 minutes, is the time where PhysiCell queries if there is any change in the cells' behavior as a result of the signaling pathway activation.

<sup>1</sup> MaBoSS is an environment for stochastic simulations of Boolean models of cell signaling networks (Stoll *et al.*, 2012, 2017).

<p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



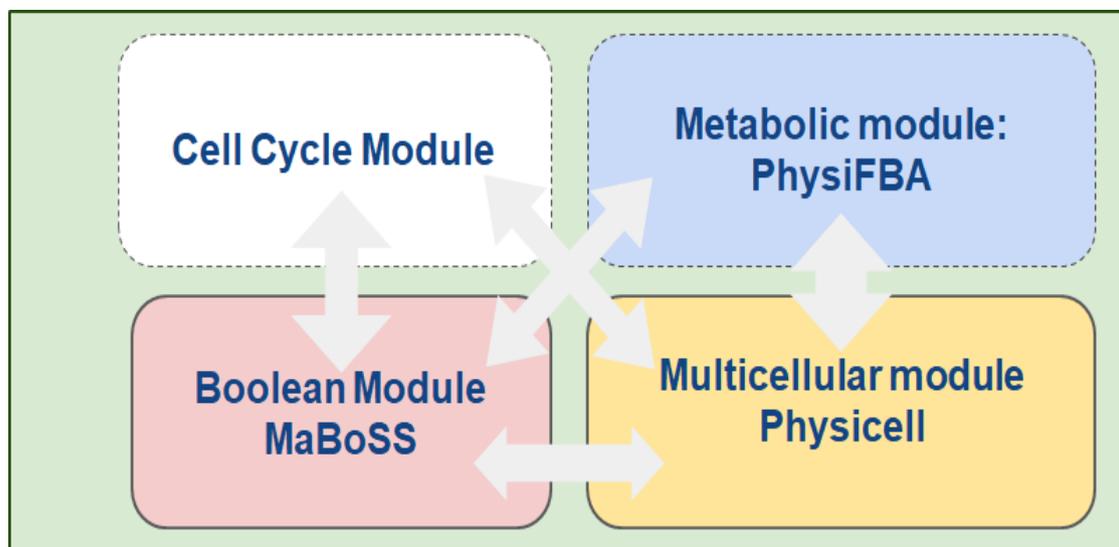
**Figure 3: Multiscale simulation main loop pseudo-code and the different time scales.**

<p>Project supported by the European Commission Contract no. 825070</p>	<p><b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b></p>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

## 4 Integration of the agent-based modeling in our multiscale Model

The agent-based model previously described was included into a multiscale modeling framework to allow us to perform the complex simulations required by the INFOR Life Science use case. This multiscale Model (MSM) frames the agent-based model (Ghaffarizadeh *et al.*, 2018; Letort *et al.*, 2018) with processes taking place at multiple time and space scales, enabling the simulation of the diffusion of chemical entities in the environment, such as oxygen or drugs; intracellular pathways, such as drug-targeted or cell cycle proteins; and cells' mechanics, such as movement and friction.

This MSM allows us to connect different scales of description and to find causal relationships among them (Figure 4). In this sense, our MSM can be divided in three differentiated components: environment, agents and signaling. The environment component is the one that simulates all the diffusion, creation and uptake of chemical entities that roam in the environment. The agents' component is the one that takes care of the population level and simulates the cells dynamics, their growth, death, movement and overall physical behavior among cells and among them and their surrounding environment. These first two components are simulated by PhysiCell. The signaling module is the one that takes care of the individual cells' level and simulates the behavior of each cell in response to its environment and its neighboring conditions. This component is simulated by PhysiBoSS. Finally, the cell cycle module that takes care of how the cells grow and divide, is embedded into PhysiCell.



**Figure 4: Scheme of the desired multiscale modeling framework. Continuous line indicates that the module has been incorporated into the framework, dotted line represents ongoing work.**

Our MSM is based on PhysiCell, a powerful lattice-free physics-based agent-based cell simulator for 2D and 3D multicellular systems (Ghaffarizadeh *et al.*, 2018) and PhysiBoSS (Letort *et al.*, 2018), a tool that merges PhysiCell with the stochastic Boolean model simulator MaBoSS (Stoll *et al.*, 2012, 2017). All of these tools are open-source and are released under free open-source licenses. As this MSM is critical to reach INFOR and its Life Science use case's objectives and KPIs, notable efforts have been made to upgrade it and scale it up.

### 4.1 The agent-based component

As described before, the population dynamics are simulated using the agent-based modeling engine from PhysiCell (Ghaffarizadeh *et al.*, 2018) (<https://github.com/MathCancer/PhysiCell>) that allows studying different physical properties' variations (cell-cell adhesions, cell-matrix adhesions) under different microenvironmental conditions (presence of oxygen, signaling molecules or extracellular matrix), as depicted in **Error! Reference source not found.** In our case, this agent-based engine has been connected to MaBoSS to simulate the signaling network of each individual agent following the structure of PhysiBoSS (Letort *et al.*, 2018), (<https://github.com/sysbio-curie/PhysiBoSS/wiki>). This allows us to have a stochastic Boolean model simulator (MaBoSS) that predicts cell fates embedded in a flexible agent-based model (PhysiCell) that simulates multicellular systems.

<p>Project supported by the European Commission Contract no. 825070</p>	<p><b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b></p>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

The different engines are connected such that the output of one is the input of another. Specifically, the availability of nutrients or the presence of drugs around a cell agent are used as inputs of the signaling model of that cell and the output of the signaling model is used to update the behavior of the cell agent. In a typical MSM simulation, each single sphere corresponds to a cell agent and the color code indicates the cellular phenotype (Figure 2). For instance, the brown-colored cells at the centre of the structure correspond to necrotic cells, *i.e.* cells that die as a consequence of the lack of nutrients and oxygen.

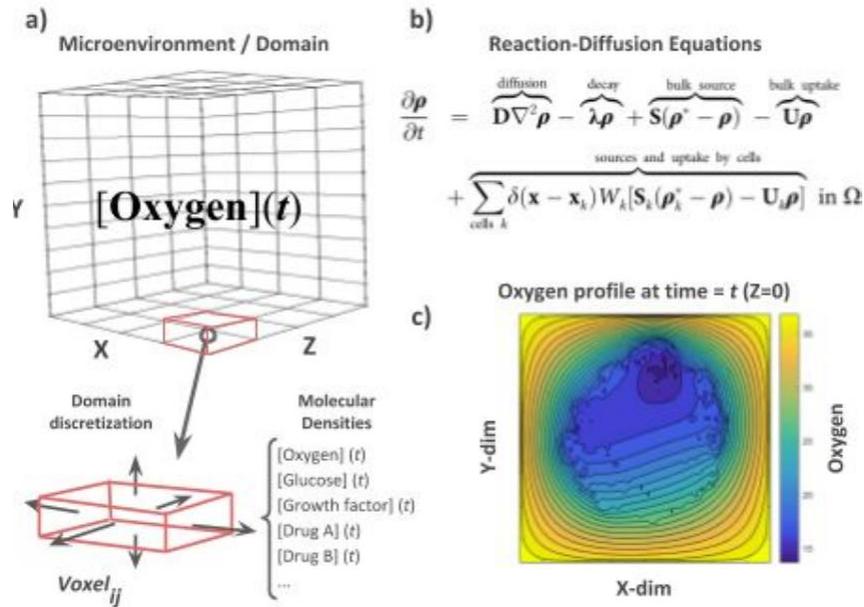
**Table 1: Parameters for cell geometry and mechanical interactions for breast epithelium MCF-7 cell line.**

Parameter	Value	Dimensions
Cell volume	2494	$\mu\text{m}^3$
Nuclear volume	540	$\mu\text{m}^3$
Cell radius	8.41	$\mu\text{m}$
Nuclear radius	5.05	$\mu\text{m}$
Cell fluid fraction	0.75	dimensionless
Heterotypic cells adhesion min	0.15	dimensionless
Homotypic cells adhesion min	0.1	dimensionless
Heterotypic cells adhesion max	0.5	dimensionless
Homotypic cells adhesion max	0.75	dimensionless
Cell motility amplitude	5	dimensionless
Cell movement persistence	0.5	dimensionless
Cell movement polarity coefficient	0.2	dimensionless
Cell cell repulsion	5	dimensionless
Cells to ECM adhesion min	2	dimensionless
Cells to ECM adhesion max	2	dimensionless

## 4.2 The environment component

The aforementioned agents are not standing in the void, instead they are communicating with a rich surrounding environment (Figure 5) with chemical entities governed by reaction-diffusion equations (Figure 5b). This environment is a dynamic one, where these entities are created, diffuse, and can be uptaken dynamically by the agents or can be added from fountains or drained from sinks, as can be seen in Table 2. For this, PhysiCell uses BioFVM, a Finite Volume Method (FVM) based simulation software, to simulate the chemical microenvironment with a vector of reaction-diffusion Partial Differential Equations (PDEs) with both bulk source/sinks and cell-centered sources and sinks (Ghaffarizadeh *et al.*, 2016).

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



**Figure 5: Domain definition, representation and simulation of the molecular densities that define the microenvironment.** a) The domain is discretized in voxels and each voxel stores the amount of each molecular densities in this space point and at a given time step. b) the systems of partial differential equations that govern the time evolution of the different densities. c) an example of the visual representation of the profile of a given density (oxygen) at the final time step for a domain of size  $1000^3$ . Bright yellow areas are substrate-producing areas and dark blue areas are areas with less substrate concentration.

In practical terms, this means that our MSM can include diffusible chemicals such as oxygen, nutrients and drugs, as well as patches of dense static cells that impede the advancement of otherwise migrating cells. These elements allow setting up complex microenvironments that enable modelers to tackle real-life scenarios like the ones found in cancer growth and migration. For instance, we can set up a scenario with different substrate-producing areas in the environment and dynamically track the substrate diffusion (Figure 5c).

**Table 2: Parameters for free-roaming chemical entities in the environment component.**

Parameter	Value	Dimensions
Oxygen initial condition	38	mmHg
Oxygen diffusion	$1 \times 10^5$	$\mu\text{m}^2/\text{min}$
Oxygen decay rate	0.1	$\text{min}^{-1}$
TNF concentration	0 - 0.5	ng / mL
TNF diffusion	1200	$\mu\text{m}^2/\text{min}$
TNF decay rate	0.0275	$\text{min}^{-1}$
Frequency of TNF injections	150	minutes
Time point at which TNF is removed from the system	1440	minutes
Duration of the regular TNF injections	10	minutes
Duration of initial injection of TNF	60	minutes
Dirichlet boundary condition	Boolean	dimensionless

### 4.3 The signalling network module

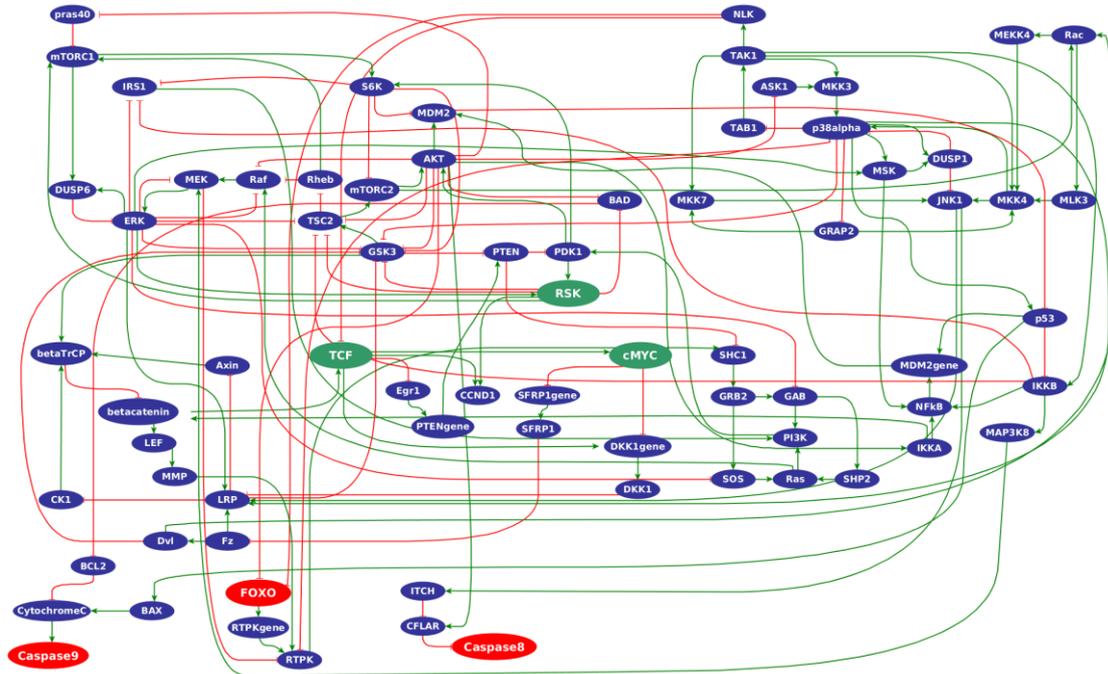
The signal transduction machinery of a cell is composed of a network of molecular components (*e.g.* protein complexes, small molecules), which allows the cell to sense external signals and adjust its internal state to respond to different stimuli (Tyson *et al.*, 2003). For instance, a schematic representation of different signaling pathways involved in prostate cancer is depicted in Figure 6. The transduction of signals like the availability of nutrients, the presence of DNA damage, etc. (Table 3) will induce the cell to respond by changing its internal state or the phenotype of the cell like moving, growing, dividing, etc.

**Table 3: Parameters for cellular processes.**

Parameter	Value	Dimensions
Replication rate	0.02	hours <sup>-1</sup>
Apoptotic rate	5.32 x 10 <sup>-5</sup>	seconds <sup>-1</sup>
Cell cycle rate	0.001	min <sup>-1</sup>
Oxygen uptake rate	32-38	attomol s <sup>-1</sup> cell <sup>-1</sup>
Oxygen sensing threshold	1	dimensionless
TNF uptake rate	0.0025	fg/cell/min
TNF secretion rate	0.1	fg/cell/min
TNF sensing threshold	1	dimensionless
Time of signaling model evaluation	10	minutes

These transductions of signals are described with signaling networks that are complex systems that exhibit non-trivial behaviors. These networks are oftentimes studied with mathematical and computational models where the nodes, representing proteins can be active or inactive and are connected through signed interactions, indicating activation or inhibition (Calzone *et al.*, 2018). The node state at a given time point is computed through an activation function, which integrates the different inputs of the node. The model state is computed by aggregating all the node states at a given time. In this project, we will simulate and analyze signaling models using the stochastic Boolean approach implemented in MaBoSS, as it allows us to compute the probabilities associated with the different phenotypes.

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h2>Deliverable D1.2</h2>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

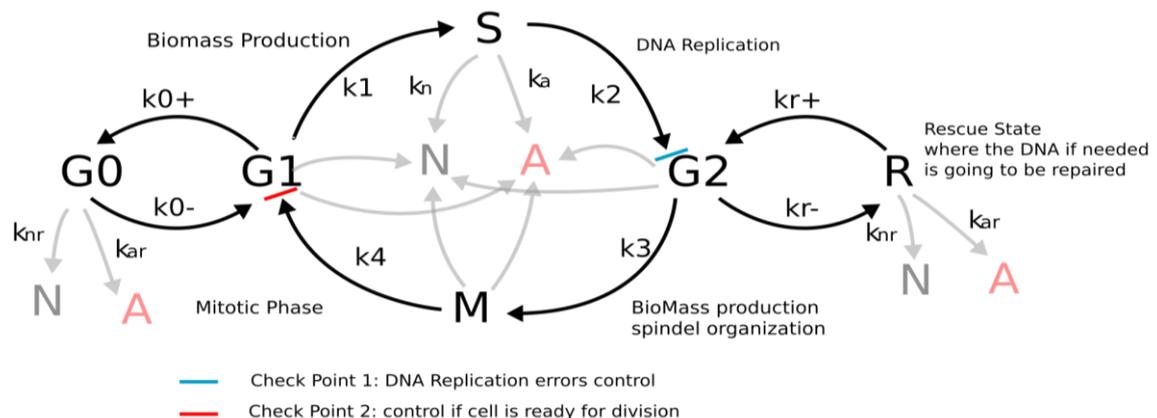


**Figure 6: Network model of the AGS cell line signaling. Nodes correspond to proteins or complexes of proteins. Red and green arrows correspond to inhibitory and activation interactions, respectively. Red and green nodes indicate those nodes associated with anti-survival and pro-survival phenotypes, respectively.**

## 4.4 The cell cycle module

In this section we present the implementation of a novel model of the cell cycle inside the simulation architecture of PhysiBoSS. This model differs from the rest of models already present in PhysiBoSS as it's more detailed, more realistic and its different phases have different cellular phenotypes that could be modeled using agents. First of all it is worthy to remember that nearly all the tumors have some kind of dysfunctionality related to the cell cycle. Indeed, this is the biochemical time regulator and it orchestrates the cellular phases during the growth. In turn, it is tuned on or off in respect to the external and internal cues. Thus, being able to correctly regulate the activity of the cell cycle is a critical feature for cancer development.

Currently, the cell cycle model implemented in PhysiBoSS is based on a set of transitions among the different cell states G1, S, G2, M (Figure 7). Each of these phases is responsible for a certain task of the growth, like biomass production, DNA replication or mitotic division. At the moment, there is no quantitative connection with the growth, indeed the cell growth rate is independent of its cellular phase.



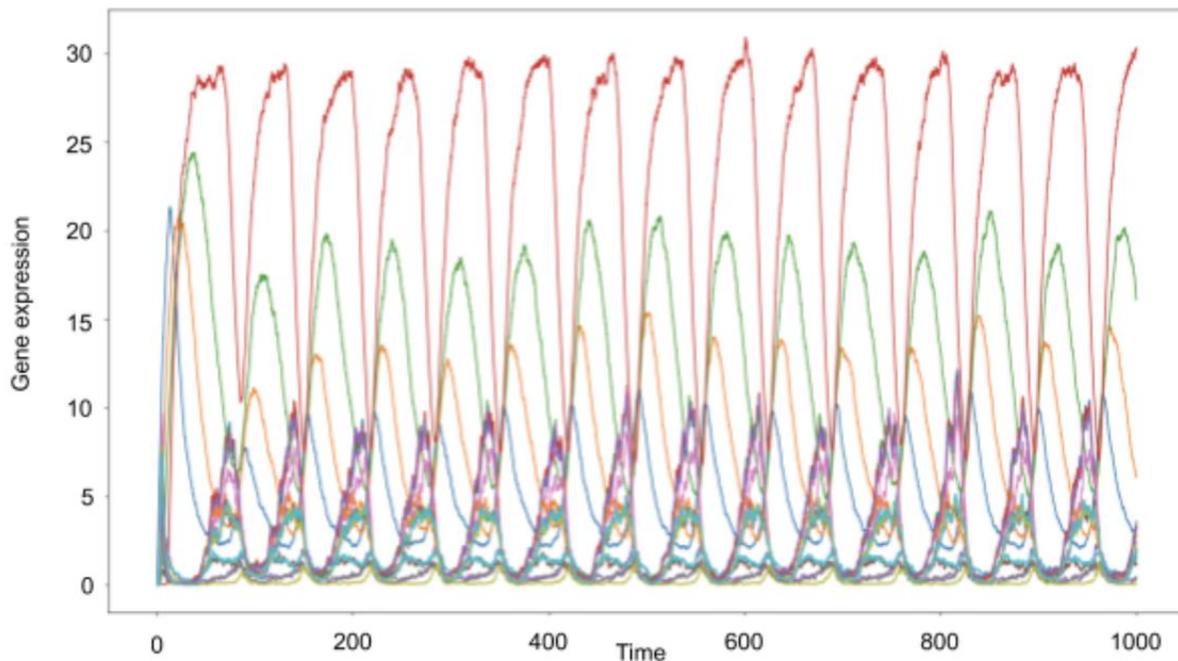
**Figure 7: Cell cycle module currently considered in PhysiBoSS.**

<p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h1>Deliverable D1.2</h1>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

Furthermore, one of the operative definitions of the cell phase is the set of genes that are expressed in that time. This means that each phase of the cell cycle can drive the metabolism in different directions that in turn will allow the cell to grow or not. In order to make the model more realistic together with the implementation of metabolic models relying on flux balance analysis (FBA) approximations, we are developing a cell cycle dynamic that can drive the genes to an oscillatory behavior, as it happens in a real cell cycle that in turn will drive the metabolism.

The model here describes the oscillatory behavior of the gene expressions and its relative transcripts during the cell cycle. First, we consider a central core of genes that are responsible for these behaviors and their checkpoints, as well as their connection with the rest of the genome. These networks have been taken from experimental and computational models from literature (Yang *et al.*, 2018, 53; Gérard and Goldbeter, 2011) and they regulate the oscillation of the genome and the checkpoint dynamic of the cell cycle. We implement a stochastic dynamic of this network (Figure 8).

We aim to connect this stochastic dynamic with the genes relative to each phase and in turn connect it to the FBA module. This connection will allow us to turn on and off the metabolism for different phases of the cell cycle building up a more realistic model of the growth dynamic.



**Figure 8: Gene expression levels oscillations driven by the cell cycle are reproduced using stochastic simulations coupled to a cell cycle model.**

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

## 5 Multiscale Model HPC Implementation

We are currently working in the deployment of the MSM simulation framework PhysiBoSS in BSC's supercomputer, the MareNostrum 4. This involves the parallelization process using different technologies such as OpenMP and MPI to scale-up our simulations in terms of numbers of cells considered. To address this complex scaling-up, we decided to first address the most basic level of simulation: the environment part and its dedicated engine, BioFVM. Our current work aims to parallelize the core kernels of BioFVM to support distributed parallelism using Message Passing Interface (MPI) enabling one to solve much larger problems with greater resolution and limited time.

BioFVM (Ghaffarizadeh *et al.*, 2016) is a Finite Volume Method (FVM) based simulation software for solving PDEs (Partial Differential Equations) that model complex processes like uptake, release and diffusion etc., of substrates for multicellular organisms. BioFVM is capable of handling multiple substrates and can simulate the biological processes such as decay, diffusion using both cell and bulk sources. The following diffusive PDE on a computational domain  $\Omega$  (and boundary  $\partial\Omega$ ) is solved for a substrate density vector  $\rho$ :

$$\partial\rho/\partial t = \nabla \cdot (D \circ \nabla\rho) - \lambda \circ \rho + f \quad (1),$$

with the boundary condition  $(D \circ \nabla\rho) \cdot n = 0$  on  $\partial\Omega$  and the initial condition  $\rho(x, t=0) = g$  in  $\Omega$ . In equation (1) above,  $D$  is the matrix of (constant) diffusion coefficients,  $\lambda$  is the decay rate,  $f$  denotes the net source term (further consisting of a bulk term and cell term) and  $\circ$  denotes the element-wise product of vectors.

BioFVM also forms the core component of PhysiCell (Ghaffarizadeh *et al.*, 2018), an agent-based multicellular system simulation software that is capable of simulating the cells' phenotypes i.e. the movement, interaction of cells etc., to be studied as a function of the diffusing substrates and signaling factors. PhysiCell is capable of handling models for cell cycle, apoptosis, necrosis etc., among many others. The design and implementation of BioFVM is such that it is scalable within a compute node, enjoys a minimum dependency on external libraries, is capable of running a multi-million cell simulation on desktops, and supports both 2D/3D simulations, among others. The software has been implemented in C++ and uses Open Multiprocessing (OpenMP) (OpenMP Architecture Review Board, 2018) to support shared memory parallelization.

The code takes advantage of extensive vectorization and the diffusion equation shown above is solved using a fast direct algorithm called the Thomas algorithm for solving a tridiagonal system of linear equations. Multiple instances of such linear systems are solved simultaneously by multiple threads that also take advantage of extensive vectorization. Such multiple instances of linear systems and solutions are made possible by splitting a higher dimensional PDE into multiple related 1-dimensional PDEs. The method that makes this splitting possible is called the locally 1-dimensional method or lod for short (Ghaffarizadeh *et al.*, 2016, 2018). Though BioFVM exhibits efficiency, reduced dependencies, ease of use, its greatest limitation is that it is limited to a single node only i.e. it is not capable of running on multiple nodes of an HPC cluster to solve a single, coherent problem. Thus, the problem size is limited by the memory of a single node. Our current work aims to parallelize the core kernels of BioFVM to support distributed parallelism using Message Passing Interface (MPI) (Message Passing Interface Forum, 2015) enabling one to solve much larger problems with greater resolution and to reduce the time to solution.

With this work, we have a two-fold aim:

1. Expose the internal design of BioFVM to aid the Scientific and HPC community to better understand the working of BioFVM and thus, be able to evaluate possible future parallelization strategies.
2. Elaborate on our design, decisions, parallelize the core kernels and present preliminary parallelization results.

We are also motivated by the fact that BioFVM forms a core component of PhysiCell (Ghaffarizadeh *et al.*, 2018). Thus, it is imperative to parallelize the core kernels of BioFVM to support distributed parallelism before PhysiCell can be parallelized. Nevertheless, BioFVM as a stand-alone software is useful enough to warrant parallelization. Further, to the best of our knowledge, no literature exists that discusses the distributed parallel design and implementation of BioFVM using MPI.

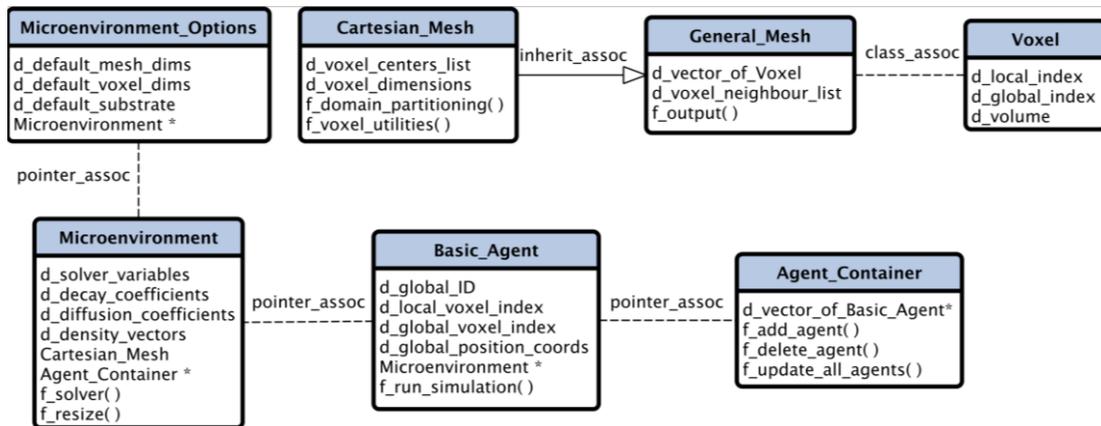
### 5.1 Design and parallelization

BioFVM supports only shared memory parallelization using OpenMP. We present the internal design of BioFVM and use MPI to parallelize the core kernels to enable it to support Hybrid parallelization (i.e. MPI + OpenMP). BioFVM supports both 2D and 3D domains and we use the latter for describing the internal design and our parallelization strategy. A 3D domain in BioFVM is divided into Voxels (Volumetric pixels). The complete, regular domain itself is

 Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

described by specifying the minimum and maximum limits of the domain in each direction. Once the dimensions of (cubic) voxels are specified, the total number of voxels contained in the domain becomes fixed. It is important to mention that in 2D, the number of voxels in the third dimension is automatically taken as one. Thus, in effect, a 2D domain can be visualized as a 3D domain with the width of the third dimension being a single voxel. We now describe the internal designs that were chosen to parallelize the core kernels of BioFVM.

The external environment in BioFVM is represented by an object of the Microenvironment class. This class declares the major entities used in the Thomas algorithm for solving a Tridiagonal system of linear equations. Further, it also contains as public members, the objects of the class Cartesian\_Mesh and a pointer to the Agent\_Container class (for brevity of space, the individual members are not being listed, but are depicted in Figure 9). At a global level, we set the name of the environment, the density of various substrates and the measurement units for the mesh space and time using an object of the Microenvironment class. Further, the most important group of functions that this class contains are the functions that calculate the number of global (homogeneous) voxels (given the dimensions of both the domain space and the voxels). The resizing functions are actually members of the Cartesian\_Mesh class mentioned above.



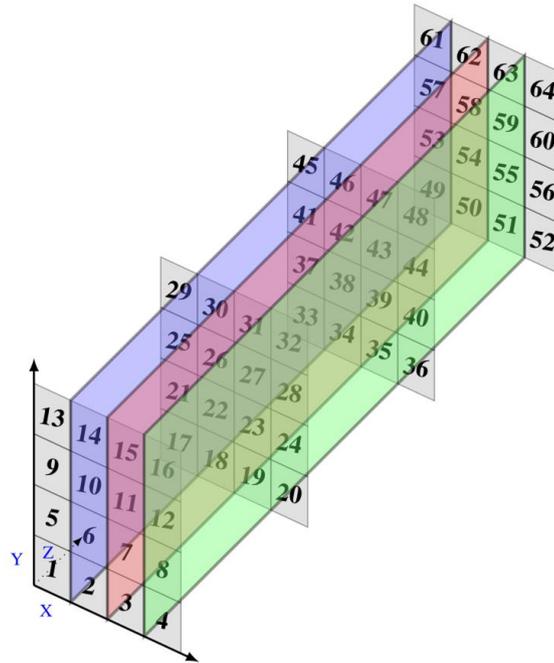
**Figure 9: Unified Modeling Language (UML) diagram of the BioFVM MPI implementation with its different classes and relationships.**

The first step in parallelization is domain decomposition or domain partitioning (Saxena *et al.*, 2016; Saxena, 2018; Saxena *et al.*, 2018) where the domain is divided into multiple smaller sub-domains and assigned to a specific MPI process. This approach of dividing the space was preferred to the alternative approach of dividing the total work instead of dividing the sub-domain as it was found difficult to quantify the total work in such a way (and since this approach leans towards a master-slave design pattern that is generally not scalable).

We assume that the x-direction (unit-stride dimension) in BioFVM goes from left to right, the y-direction going from bottom to top and the z-direction going inwards. These directions are different from the directions assumed for a 3D MPI Cartesian topology i.e. the MPI Cartesian topology assumes the x-direction going from top to bottom, the y-direction going from left to right and the z-direction going inwards. This assumption of directions is important:

1. to establish a consistent terminology,
2. to visualize the partitioning of sub-domains and
3. to derive the global index of the voxels on each MPI process.

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h1>Deliverable D1.2</h1>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



**Figure 10:** A 3D domain of dimensions  $4 \times 4 \times 4$  can be visualized as four 2D plates of dimension  $4 \times 4$  arranged one after the other. A pure 1D domain partition of 4 MPI processes in the x-direction divides the voxels numbered from 1 to 64 into 4 parts. Rank 0 process contains voxel IDs  $4n+1$ , rank 1 process contains voxel IDs  $4n+2$ , rank 2 process contains voxel IDs  $4n+3$  and rank 3 process contains voxel IDs  $4n+4$ , where  $n = 0, 1, 2, \dots, 15$ . Data is contiguous in the x-direction and the distance between 2 consecutive elements in the y and z direction is 4 and 16, respectively.

Furthermore, a pure x-decomposition refers to the division of the x-direction of BioFVM among multiple processes. It can now be noted (according to our assumptions) that creating a pure x-decomposition requires creating a 1D MPI topology that has processes only in the y-direction. Thus, if P is the total number of MPI processes,  $\text{dims}[3]$  represents MPI processes in the x, y and z direction, respectively, we set  $\text{dims}[0]=1$ ,  $\text{dims}[2]=1$  and  $\text{dims}[1]=P$  to obtain a pure x-decomposition in BioFVM. To further clarify, the stride between consecutive elements in a direction increases in the order  $z > y > x$ , with x being the unit-stride dimension. The impact of a good domain decomposition on the performance of the diffusion solver in BioFVM cannot be emphasized enough and it's further detailed in the following sections. For simplicity, we use a 1D MPI Cartesian topology of processes instead of a 3D topology to divide the physical domain into sub-domains although the mapping functions between MPI ranks, voxel indexes and points in 3D space apply to general 3D decompositions in 3D space (Figure 10).

Before a domain decomposition is performed, each MPI process initializes an object of the Microenvironment class. Further, each process in our design maintains the local and global values of the number of voxels, local voxel indexes ( $\text{mesh\_index}$ ), global mesh index of the local voxels and the center of each local voxel's global coordinates. The group of resizing functions that perform the domain partitioning are members of the Cartesian\_Mesh class mentioned above. Algorithm 1 shows how the domain partitioning algorithm assigns voxels to each MPI process. First the global dimensions of the domain and the voxel dimensions are used to decide the number of local voxels. This is followed by the calculation of the global coordinates of the centers of voxels (for brevity lines 1-6 in Algorithm 1 only show this for the x-direction and y- and z-direction are analogous). Further, since each voxel must maintain the local and global mesh index, a local start of the global mesh index is calculated on each process, which then is used to assign the global mesh index to each voxel (see the triple nested loop in Algorithm 1). Apart from the domain decomposition, a list of the immediate directional neighbors of each voxel is also maintained (not shown in Algorithm 1). Such a scheme must accommodate for the cases when there is no local left/right neighbor or when the process is aligned to the physical boundary of the domain. In the serial BioFVM, a list for the Moore neighborhood is also built for each voxel but since we do not find any examples that utilized this neighborhood, we abstain from parallelizing this routine. Note that the Moore neighborhood equates to a 9-point stencil in 2D and a 27-point stencil in 3D (Kamil *et al.*, 2010).

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

The Basic\_Agent class forms an abstraction of a cell. An object of the Basic\_Agent class can either act as a source, secreting a substrate or as a sink, uptaking a substrate. Whenever an object of this class is created, the secretion rate, uptake rate and the saturation densities are set to a value of zero. Each agent has an ID, a unique index having a value between zero and the total number of agents created, a type (source or sink), and a volume and maintains the local index of the voxel it is currently present in. In a distributed environment, we added a data member to represent the global index of the voxel where the agent currently resides. Furthermore, an agent's position in space is represented by x, y and z coordinates that are generated randomly. It is to be noted that the ID of an agent is unique throughout the physical domain and hence if multiple threads of a single MPI process are being used to create agents, then the increment of the ID must be made thread-safe. Similarly, if multiple MPI processes are used to create agents, no two processes can use the same ID to the agent that they create.

**Require:**  $xmin, xmax, d[]$  (Topology Dimensions),  $c[]$  (Process Coordinates),  $\Delta x$  (Voxel x-length)

```

1:  $g\_x\_nodes \leftarrow \frac{(xmax-xmin)}{\Delta x}$   $\triangleright$   $y$  and  $z$  analogous
2:  $l\_x\_nodes \leftarrow \frac{g\_x\_nodes}{d[1]}$ 
3:  $l\_x\_start \leftarrow xmin + (c[1] \times l\_x\_nodes \times \Delta x)$ 
4:  $i \leftarrow 0$ 
5: while  $i++ \leq l\_x\_nodes - 1$  do
6:    $x\_c[i] \leftarrow l\_x\_start + (i + 0.5) * \Delta x$ 
7:  $x_l \leftarrow (d[0] - c[0] - 1) \times x\_nodes \times l\_y\_nodes$ 
8:  $y_l \leftarrow c[1] \times l\_x\_nodes$ 
9:  $z_l \leftarrow c[2] \times x\_nodes \times y\_nodes \times l\_z\_nodes$ 
10:  $l\_strt\_g\_index \leftarrow x_l + y_l + z_l$ 
11:  $n, z, y, x \leftarrow 0$ 
12: while  $k++ < l\_z\_nodes$  do
13:    $z_k \leftarrow k \times g\_x\_nodes \times g\_y\_nodes$ 
14:   while  $j++ < l\_y\_nodes$  do
15:      $y_j \leftarrow j \times g\_x\_nodes$ 
16:     while  $i++ < l\_x\_nodes$  do
17:        $vxl[n].cntr[0] \leftarrow x\_c[i]$ 
18:        $vxl[n].cntr[1] \leftarrow y\_c[j]$ 
19:        $vxl[n].cntr[2] \leftarrow z\_c[k]$ 
20:        $vxl[n].g\_indx \leftarrow l\_strt\_g\_index + z_k + y_j + i$ 
21:        $n \leftarrow n + 1$ 

```

**Algorithm 1: Assignment of voxels to MPI processes in 1D x-Domain Decomposition. Only partitioning of x-dimension is shown (y and z-directions are analogous). Prefixes l and g stand for "local" and "global", respectively. Array d[] contains the MPI cartesian topology dimensions and the array c[] MPI process coordinates (Message Passing Interface Forum, 2015). The triply nested loop sets the global voxel (vxl) centers (cntr) and the global voxel index (indx).**

BioFVM uses the Thomas solver (Thomas, 1949) for solving a tridiagonal system of linear equations that result from the FVM discretization of diffusion PDEs. This solver is inherently serial and hence cannot be fully parallelized. However, there do exist parallel algorithms capable of solving tridiagonal systems of linear equations but with an increase in operation count and significant complexity of implementation (László, 2016). Thus, we perform the

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



domain decomposition in only 1 direction. This makes the solver completely parallel in two directions but sequential in the third direction.

Additionally, the pure OpenMP version of BioFVM supports only the serial writing of the result data such as the concentration of the substrate after a specified time interval in a .mat file. In our Hybrid implementation, instead of gathering data at the root process, we use MPI-IO so that processes can view their portion of the file and write their part of the data simultaneously. BioFVM first writes a 20-byte fixed header into the output file and then the three dimensional centers of voxels, the voxel volume and the concentration of the substrates in that voxel for each substrate. For all our experiments, we use the MareNostrum IV (MN4) supercomputer at the Barcelona Supercomputing Center (BSC). There are a total of 3456 nodes where each node has two Intel Xeon Platinum 8160 processors with a base frequency of 2.1 GHz. Each of the processors have 24 cores and the cores in each processor share a main memory of 48 GB. The computer nodes are interconnected using the Intel Omni-Path (OPA). The Operating System (OS) that the cluster uses is the SUSE Linux Enterprise Server 12 SP2. Further, we use GCC 8.1 and OpenMPI 3.1.1 as our compiler and MPI implementation respectively. We chose GCC as the compiler because the user documentation of BioFVM takes the GCC as the “gold standard”.

## 5.2 Experiments

Since the addition of MPI to OpenMP results in a Hybrid code i.e. MPI+OpenMP, after empirical evaluation we choose the hybrid decomposition for all our experiments as  $2 \times 24$  i.e. we spawn a total of 2 MPI processes per node (1 process for each socket or processor) where each MPI process spawns 24 OpenMP threads. This choice is also advocated by the existing literature (Smith and Bull, 2001; Rabenseifner *et al.*, 2009) as it eliminates the unwanted overhead of the first touch policy (Chandra *et al.*, 2001; Chapman *et al.*, 2008) as there are parts of the application where initializations are carried out with a single (master) thread. Nevertheless, we explicitly mention whenever we use any other Hybrid configuration of MPI processes and OpenMP threads. We emphasize that, for performance reasons, the threads must be bound to the individual cores of the socket (Pas *et al.*, 2017).

Further, in all our tests, the domain and the voxel dimensions remain cubic (without any loss of generality). Thus, for example, if the domain is  $1000 \times 1000 \times 1000$  and the voxel dimensions are  $10 \times 10 \times 10$ , then there exist 100 voxels in each direction, making it a total of  $10^6$  (1 million) voxels in the physical domain. Without any loss of generality, we assume that the dimensions of the physical domain is exactly divisible by the dimension of the voxel in that direction.

The example that we choose to implement and demonstrate the benefits of Hybrid parallelism is tutorial1 in the BioFVM/examples directory (<http://www.mathcancer.org/blog/biofvm-tutorials/>). Our aim was to completely parallelize the execution path of tutorial1, as well as, the core components of BioFVM, such as partitioning the domain, creation of Basic Agents and writing of the result file, etc. The example under consideration does the following:

1. It initializes and resizes the microenvironment (abbreviated as microenvironment ) of BioFVM,
2. it creates a Gaussian profile of the concentration of a substrate,
3. it writes the initial concentration to a .mat file,
4. it creates Basic Agents (Sources + Sinks),
5. it simulates Sources/Sinks and Diffusion, and
6. it writes the final concentration to a .mat file.

We measure and compare the execution time of each of the steps above in the given pure OpenMP and our Hybrid implementation (i.e. MPI + OpenMP).

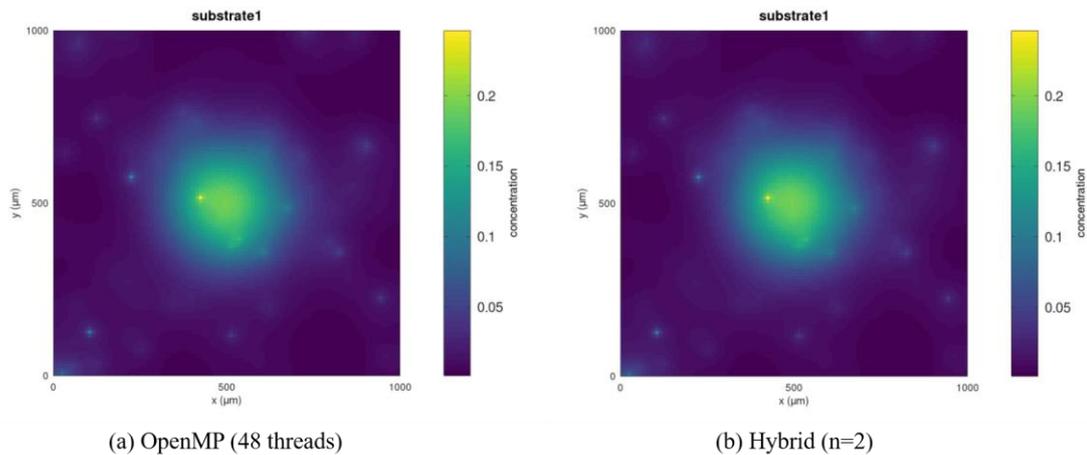
We made a design decision in parallel settings to generate the random positions of agents (both sources and sinks) on the root process (process with MPI rank zero) and then convey these positions to the appropriate processes depending on the region of space that the processes manage according to the domain partition. This was done to maintain the “random” nature of position generation. The alternative of assigning each process an equal number of agents takes away the random nature as there would be an equal number of agents in each sub-domain. This demonstrates a trade-off of choosing between parallelization in generation and maintenance of randomness.

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h1>Deliverable D1.2</h1>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

**Table 4: Time in seconds of the execution for the OpenMP version and the Hybrid versions of tutorial1 of BioFVM with a pure 1D x-decomposition. The total voxels are  $10^6$  and each voxel volume is  $10^3$ . The pure OpenMP version uses 48 threads. Hyb (n=a) means we run on 'a' nodes where we have 2 processes per node and 24 threads per process by default. Hyb (n=5) (10 MPI processes) was needed with this domain as Hyb (n=4) (8 MPI processes) produced a divisibility problem.**

1000x1000x1000	OpenMP	Hyb(n=1)	Hyb(n=2)	Hyb(n=5)
Build $\mu$ -environment	1.14	1.03	0.51	0.20
Gaussian Profile	0.034	0.012	0.005	0.006
Initial File Write	0.223	0.110	0.120	0.207
Agent generation	0.001	0.0009	0.002	0.0009
Source/Sink/Diffusion	1.927	2.77	2.68	5.34
Final File Write	0.231	0.052	0.091	0.17
Total Time	3.56	3.98	3.41	5.93

We present results for the parallelized tutorial1 example on domains of sizes  $1000^3$ ,  $1920^3$  and  $3840^3$  and cubic voxels having a volume of  $10^3$ ; all the results were done using the pure 1D x-decomposition. While increasing the number of nodes, the communication time becomes a limiting factor for a high number of nodes (or cores) and thus the performance starts deteriorating after a certain number of MPI processes or nodes are reached (due to the Strong Scaling). To obtain the total number of executing threads, we can use the following expression:  $n * ppn * tpp$ , which represents the number of nodes ( $n$ )  $\times$  processes per node ( $ppn$ )  $\times$  threads per process ( $tpp$ ). We denote the Hybrid implementation as "Hyb( $n=a$ )", where "a" denotes the total number of nodes and the original pure OpenMP implementation as simply "OpenMP". For example, with Hyb( $n=2$ ), we obtain a total of  $2 \times 2 \times 24 = 96$  executing threads. The number of Basic Agents generated for the results presented in Table 4, Table 5 and Table 6 are 1000 i.e. 500 sources and 500 sinks. It is to be noted that for the domain of size  $1000^3$ , due to a divisibility problem with 8 processes, we used 10 MPI processes i.e.  $n=5$  nodes with 2 processes-per-node ( $ppn$ ) with 24 threads-per-process ( $tpp$ ), using 2 MPI processes-per-node (or 1 MPI process-per-socket).



**Figure 11: Final concentration plot of a substrate using (a) pure OpenMP (b) Hybrid MPI + OpenMP for a domain of size  $1000^3$  and corresponding to Table 4 to check the correctness of code.**

To check the correctness of our Hybrid code for the chosen example (for the run in Table 4), the concentration of the diffusing substrate was plotted for the pure OpenMP (Figure 11a) as well as for the Hybrid code (Figure 11b), where it can be seen clearly that the final concentration plots are identical. We abstain from plotting the concentration plots to illustrate the correctness of the implementation, unless the plot necessarily conveys further information and is needed. Table 5 shows the performance results for a domain of size  $1920 \times 1920 \times 1920$  and a voxel size of  $10 \times 10 \times 10$ . Increasing the resolution further, we test our distributed BioFVM kernels using a domain size  $3840 \times 3840 \times 3840$  and its results can be studied in Table 6.

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h1>Deliverable D1.2</h1>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

**Table 5: Time in seconds of the execution for the OpenMP version and the Hybrid versions of tutorial1 of BioFVM with a pure 1D x-decomposition. The total voxels and resolution are  $\approx 7$  million. The pure OpenMP version uses 48 threads and Hyb(n=a) translates to  $a \times 2 \times 24$  threads in all.**

<b>1920x1920x1920</b>	OpenMP	Hyb(n=1)	Hyb(n=2)	Hyb(n=4)
Build $\mu$ -environment	8.97	7.90	3.86	1.88
Gaussian Profile	0.112	0.069	0.035	0.018
Initial File Write	1.53	0.35	0.26	0.28
Agent generation	0.0013	0.0009	0.0007	0.0013
Source/Sink/Diffusion	18.96	23.65	22.49	26.25
Final File Write	1.55	0.51	0.35	0.23
Total Time	31.11	32.49	26.99	28.67

**Table 6: Time in seconds of the execution for the OpenMP version and the Hybrid versions of tutorial1 of BioFVM with a pure 1D x-decomposition. The total voxels and resolution are  $\approx 56$  million. The pure OpenMP version executes using 48 threads and Hyb(n=a) translates to  $a \times 2 \times 24$  threads in all.**

<b>3840x3840x3840</b>	OpenMP	Hyb(n=1)	Hyb(n=2)	Hyb(n=4)
Build $\mu$ -environment	77.85	68.09	32.47	16.34
Gaussian Profile	0.79	0.55	0.28	0.14
Initial File Write	12.42	1.85	1.64	1.78
Agent generation	0.0015	0.0020	0.0016	0.0008
Source/Sink/Diffusion	175.2	193.9	152.7	169.6
Final File Write	13.80	2.10	2.29	1.63
Total Time	280	266.5	189.4	189.5

With a problem of size  $7680^3$ , the memory consumption of the pure OpenMP program reaches  $\approx 97\%$  and it throws a bus error due to hitting the memory limit. Thus, an input problem of size  $7680^3$  cannot be executed on a single node of MN4. Irrespective of the performance, parallelization becomes necessary for problems that cannot fit into the working memory of a single node. A problem of size  $7680^3$  equates to approximately 452 million voxels when the voxel dimension is  $10^3$ . We executed the same problem in parallel on two nodes using our Hybrid MPI implementation but the execution terminated by giving a memory error. Since we suspected that the memory was a problem, we executed the Hybrid code on 8 nodes i.e. using  $8 \times 2 \times 24 = 384$  threads in all. The application executed successfully as can be seen in Table 7.

**Table 7: Time in seconds of the execution for the OpenMP version and the Hybrid versions of tutorial1 of BioFVM with a pure 1D x-decomposition. The total voxels and resolution are  $\approx 500$  million. The successful runs with 8 nodes and a minimum of 4 nodes use 384 and 192 threads respectively.**

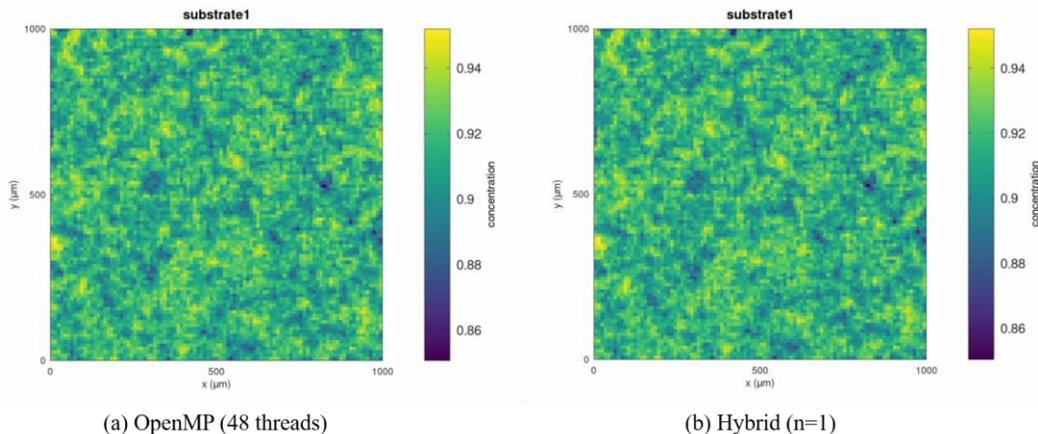
<b>7680x7680x7680</b>	OpenMP	Hyb(n=8)	Hyb(n=4)
Build $\mu$ -environment	-	67.81	141.98
Gaussian Profile	-	0.448	0.916
Initial File Write	-	4.1	2.56
Agent generation	-	0.0023	0.1060
Source/Sink/Diffusion	-	1210.41	1109.69
Final File Write	-	3.32	4.83
Total Time	-	1286.1	1260

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

**Table 8: Time in seconds of the execution for the OpenMP version and the Hybrid versions of tutorial1 of BioFVM with a pure 1D x-decomposition. The total voxels are  $10^6$ . The pure OpenMP version utilizes all 48 threads of a node and the Hybrid version spawns 1 MPI process per socket/processor.**

1000x1000x1000	OpenMP	Hyb(n=1)
Build $\mu$ -environment	1.14	1.03
Gaussian Profile	0.0157	0.0117
Initial File Write	0.219	0.084
Agent generation	2.46	1.45
Source/Sink/Diffusion	7.48	5.88
Final File Write	0.22	0.063
Total Time	11.56	8.54

To test a different case, we ran a simulation on a domain of size  $1000^3$  but increased the number of Basic Agents to  $2 \times 10^6$  (sources + sinks). As can be seen from Table 8, there is a gain of about 26-30% using Hybrid MPI even on a single node. The reason for the gain is that the generation of and the computation on agents are being simultaneously carried out on two separate processes in the Hybrid implementation as opposed to a single thread in the pure OpenMP implementation. The correctness of the execution can be verified from the two identical figures i.e. Figure 12a and Figure 12b corresponding to the pure OpenMP version and the Hybrid version, respectively.



**Figure 12: Final concentration density of a given substrate with  $2 \times 10^6$  agents on a domain of size  $1000^3$  using (a) Pure OpenMP (b) Hybrid MPI + OpenMP.**

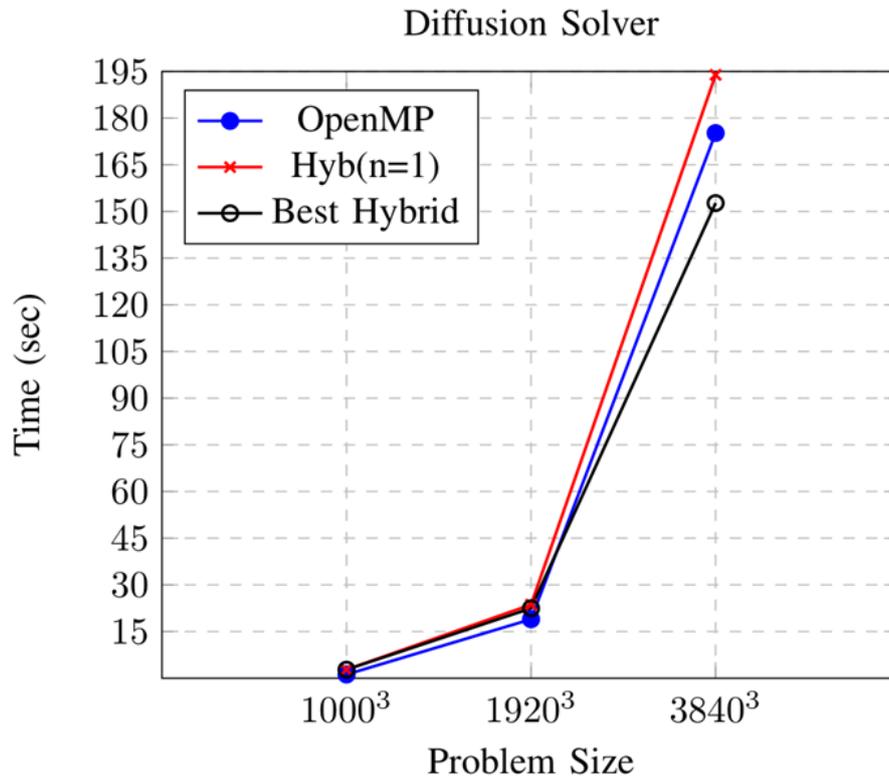
### 5.3 Discussion

BioFVM enables the simulation of biological processes such as secretion, uptake, and diffusion for multicellular organisms. Internally it sets-up a microenvironment and uses PDEs to represent the biological processes. After discretization using the Finite Volume Method (FVM), these PDEs are numerically solved using a direct solver for the tridiagonal system of linear equations. Currently, BioFVM suffers from a serious limitation as it only supports shared memory parallelization using OpenMP and thus the size of the problems that it can solve is limited by the memory of a single node. With the aim to remove this limitation, we restructured the base data-structures and functions of BioFVM to add support for distributed parallelism using MPI.

To provide a proof of concept, we present the parallelization of a chosen example and in the process, successfully parallelize the key kernels of BioFVM. For instance, we implement a pure 1D x-direction MPI Cartesian Topology and assign sub-domains to individual processes, generate basic agents that represent cells on the root process and map the positions to processes that these basic agents belong to, and parallelize the writing of result files among many other changes.

<p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

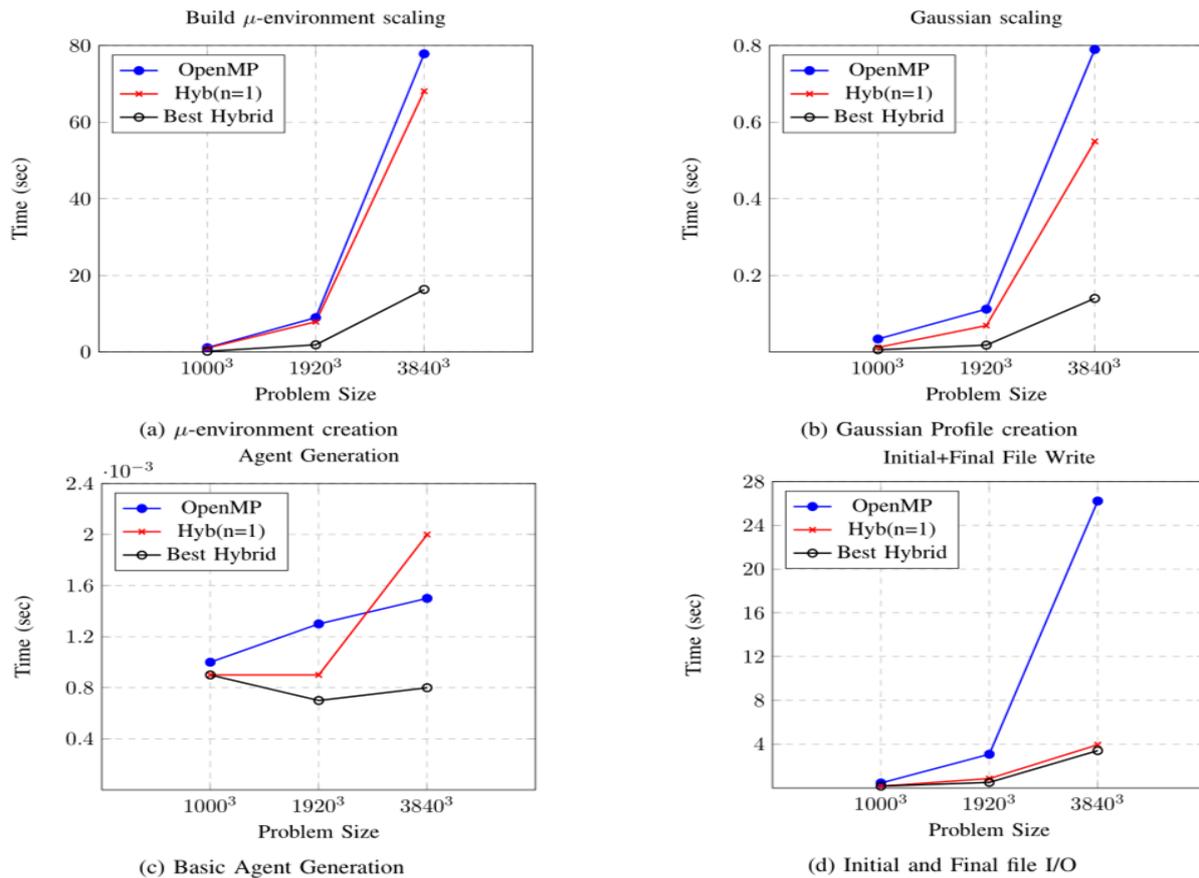
Despite the fact that the solver is only 2/3 parallel and communication bound, we see a gain of about 4.21% over the pure OpenMP implementation with Hyb (n=2) for a (small) domain of size  $1000^3$  and with 1000 Basic Agents (Figure 13). Note that it is very difficult to beat the performance of the pure OpenMP implementation on a small problem size. As seen in Figure 14, one of the reasons for this gain is the excellent scaling of the phases: (1) creating a microenvironment, (2) generating a Gaussian profile and (3) the agent generation phase. Furthermore, using MPI-IO we substantially reduced the time for writing the files. With a domain of size  $1920^3$  and 1000 basic agents we see a performance gain of 13.24% with Hyb (n=2) and this gain approximately grows to 33% for a domain of size  $3840^3$ . A domain of size  $7680^3$  cannot be executed on a single node as the application runs out of memory but was successfully executed using 4 and 8 nodes. As an alternate test, we increased the number of agents to  $2 \times 10^6$  on a domain of size  $1000^3$  and observed a performance gain of 26.12% over the pure OpenMP version.



**Figure 13: Execution time for the Thomas algorithm using pure OpenMP and Hybrid implementation. The Hybrid solver is completely parallel in the y and z direction but only on-node parallel in the x-direction.**

We see a high performance gain in Basic Agent generation, building the microenvironment, file I/O (Figure 13) but a sub-optimal gain in the diffusion solver as the solver remains partially parallelized (Figure 14). Most importantly, we expose the structure of BioFVM to evaluate parallelization schemes and make it possible to simulate larger and more complex problems i.e. BioFVM simulations can use multiple nodes and are not in any way limited to the memory of a single node.

 Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



**Figure 14: Pure OpenMP execution times with increasing problem size Vs Hybrid MPI times for the same. Hyb(n=1) represents the time when a single node with 2 MPI processes and 24 threads is used. The Best Hybrid represents the least time for that kernel for any number of experimental nodes considered. The kernels considered are: (a) Creating the microenvironment (b) Building the initial Gaussian profile (c) Generation of Basic Agents (d) Sum of the initial and final I/O file write.**

BioFVM constitutes the core at the center of PhysiCell and its parallelization is a required milestone to address the parallelization of PhysiCell. We have exposed the design of the internals of BioFVM, describe our parallelization strategy and present preliminary results. Next, we plan to work towards optimally parallelizing PhysiCell/BioFVM using an object-oriented framework, a methodology consistent with the object-oriented designs of BioFVM and PhysiCell.

Furthermore, since the solver is only parallelized using OpenMP, we aim to make it fully parallel by exploring parallel tridiagonal solver algorithms such as recursive doubling etc. As PhysiCell allows the cells to move across subdomains, a strategy is needed to asynchronously track the movement of cells across subdomains. This poses an additional challenge in terms of choosing an appropriate design pattern and a possibility of correctly using the MPI Remote Memory Access (RMA) operations. Future implementations can use 3D MPI Cartesian topologies instead of a 1D topology that we used in the current work, if there is a significant performance gain. A plethora of literature advocates the three dimensional decomposition for 3D decompositions can reduce the total volume of data that is exchanged but there exists literature that refutes these claims (Saxena *et al.*, 2018).

The MPI-ready BioFVM code is freely available from [https://gitlab.bsc.es/gsaxena/physicell\\_x](https://gitlab.bsc.es/gsaxena/physicell_x) and the ongoing efforts to have an MPI-ready PhysiCell, also termed DistPhy, are available from <https://gitlab.bsc.es/gsaxena/distphy>.

<p>Project supported by the European Commission Contract no. 825070</p>	<p>WP1 T1.1-T1.3 Deliverable D1.2</p>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

## 6 Extreme model exploration with optimization algorithms

Modern simulation-based application studies consist of large numbers of simulations with many possible variations. Simulations may be run with different parameters, possibly as part of an automated model parameter optimization, classification, or, more generally, model exploration (ME). Constructing the software to run such studies at the requisite computational scales is often unnecessarily time-consuming and the resulting software artifacts are typically difficult to generalize and package for other users (Ozik *et al.*, 2016). Applying ME to MSMs involves an iterative workflow where simulations are run across a high dimensional parameter space and changing initial conditions to explore alternative simulation outcomes.

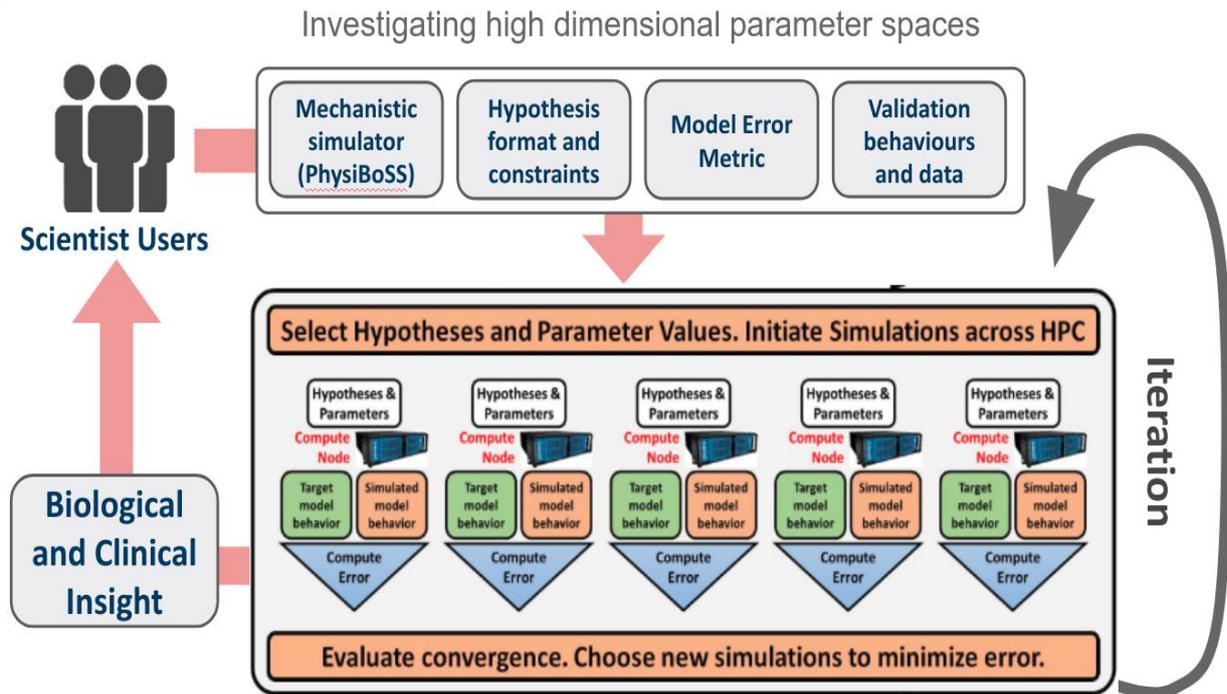


Figure 15: Model Exploration workflow adapted from (Ozik et al, 2018).

In a typical ME workflow, simulations’ outputs from a set of in-silico experiments are evaluated against some predetermined metric, which informs the next iteration of simulation experiments (Figure 15). This metric is problem-specific and can be as simple as a linear fit or as complicated as a reinforced learning algorithm. In fact, the ME process can be enhanced with the use of complex event forecasting techniques, which can be used to improve the parameter space exploration.

Here, we present some preliminary studies on the use of different metrics to find sets of parameters that reproduce a desired simulation’s behavior.

### 6.1 Model exploration of simulation parameters

The first part of the model exploration strategy focuses on performing different simulations with slightly different parameters using the EMEWS framework. These different parameter sets are obtained by sweeping their values according to parameter-specific ranges. Extreme-scale Model Exploration with Swift/T (EMEWS), uses the general-purpose parallel scripting language Swift (Armstrong *et al.*, 2014) to generate highly concurrent simulation workflows. These workflows enable the integration of external ME algorithms to coordinate the running and evaluation of large numbers of simulations. The general-purpose nature of the programming model allows the user to supplement the workflows with additional analysis and post-processing as well (Ozik *et al.*, 2016). EMEWS is particularly useful as it offers the following contributions to the science and practice of simulation ME studies:

<p>Project supported by the European Commission Contract no. 825070</p>	<p><b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b></p>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

1. it offers the capability to run very large, highly concurrent ensembles of simulations of varying types;
2. it supports a wide class of model exploration algorithms, including those increasingly available to the community via Python and R libraries; and
3. it offers a software sustainability solution, in that simulation studies based around EMEWS can easily be compared and distributed.

EMEWS framework and its high-throughput hypothesis testing has already been applied to the complex problem of tumor-immune interactions integrating it with PhysiCell and BioFVM (Ozik *et al.*, 2018, 2019).

This tumor-immune interactions model is a simple model of 3D immunosurveillance against heterogeneous tumors, with a special focus on the spatial dynamics of stochastic tumor-immune contact interactions that was described in detail in (Ghaffarizadeh *et al.*, 2018). In this model, each cancer cell has a mutant “oncoprotein” which drives proliferation: the greater the expression of  $p$ , the more likely the cell grows and divides, but also has higher immunogenicity.

To model immunosurveillance, after simulating 14 days of growth Ghaffarizadeh *et al.* introduced generic immune cell agents that move towards tumor cells by chemotaxis, stochastically form adhesions to any cell in close contact, and then test for immunogenicity by attempting to induce apoptosis with a probability that scales linearly with immunogenicity. If successful, the tumor cell undergoes apoptosis, while the immune agent detaches and resumes its chemotactic search for additional tumor cell targets. If the immune cell does not kill the tumor cell, it remains attached while making further attempts to induce apoptosis until either succeeding or reaching a maximum attachment lifetime, after which it detaches without inducing apoptosis. Using this model, researchers studied the changes in the model’s behaviors upon sweeping six parameters (Table 9).

**Table 9: Parameter space of the workflow from (Ozik *et al.*, 2018).**

Parameter	Min	Max	Increment	Dimensions
Immune cell apoptosis rate	$6.94 \times 10^{-6}$	$6.94 \times 10^{-4}$	$8.5882 \times 10^{-5}$	$\text{min}^{-1}$
Oncoprotein threshold	0.1	1	0.1125	dimensionless
Immune cell kill rate	0.1	1	0.1125	$\text{min}^{-1}$
Immune cell attachment rate	0.01	1	0.12375	$\text{min}^{-1}$
Immune cell attachment lifetime	10	90	10	minutes
Immune cell migration bias	0.1	0.9	0.1	dimensionless

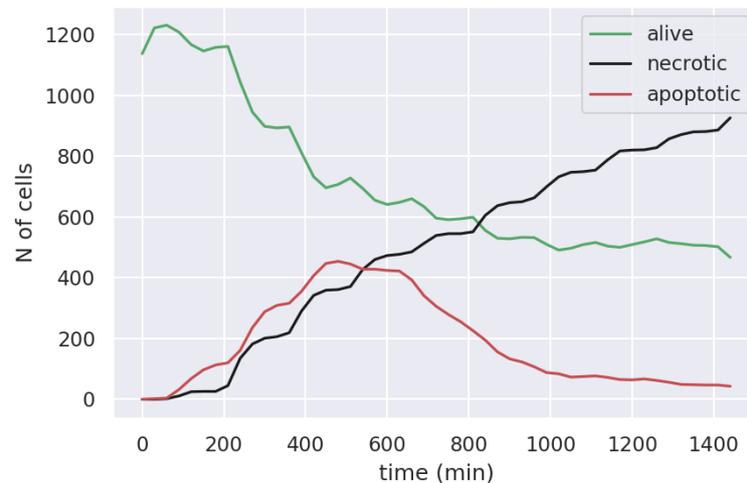
We have adapted the EMEWS framework to work with PhysiBoSSa and the study of TNF-dependent behavior. This study uses a signaling pathway model consisting of a network of 31 nodes that describes the behavior of cells in response to different TNF regimes. In response to TNF presence, the cells can Proliferate, die by Apoptosis or Necrosis or remain in a Naive state of survival.

This study was already performed in PhysiBoSS (Letort *et al.*, 2018), however we reproduce it in the upgraded version of PhysiBoSSa, where many of the functions, their parameters and their implementation have changed, more notably the internalization of free-roaming environmental chemical entities into the agents and how they trigger effects in the signaling pathways. Also, we wanted to study the heterogeneity of the parameter space by finding the sets of parameters that evolve in the same way; that is, different sets of parameters that give the same global simulation: have a depletion of proliferative cells. Thus, we focused on finding a proper set of values for these modified parameters (Table 10) that allow for the reproduction of the simulations in which proliferative cells are depleted upon the addition of TNF each 150 minutes (Figure 16).

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

**Table 10: Parameter space of the workflow for the TNF-dependent behavior example.**

Parameter description	Set of parameters	Min	Max	Increment	Dimensions
Concentration of TNF	NCSR	0	100	5	ng / mL
Frequency of TNF injections	NCSR	0	1440	10	minutes
Time point at which TNF is removed from the system	NCSR	0	1440	10	minutes
Duration of the TNF injections	NCSR	1	200	10	minutes
Oxygen sensing threshold	NCSR	0.1	0.5	0.01	dimensionless
TNF sensing threshold	BSC	0.1	0.5	0.01	dimensionless
TNF uptake rate	BSC	0	2	0.01	ng/mL/voxel/min
TNF secretion rate	BSC	0.01	1	0.01	ng/mL/voxel/min
Cell cycle rate	BSC	0.0001	0.01	0.0001	min <sup>-1</sup>
Time of signaling model evaluation	BSC	5	20	0.5	minutes
Duration of initial injection of TNF	BSC	1	25	1	minutes



**Figure 16: Spheroid's dynamical changes to pulses of TNF injection (0.5 ng/mL during 10 minutes) each 150 minutes. Green, Proliferative cells; red, Apoptosis; black, Necrosis. Initial spheroid radius is 100 mm.**

## 6.2 Use of different optimization metrics to find proper sets of parameters

The second part of the model exploration strategy involves the use of an optimization metric that evaluates and ranks the performance of the different parameter combinations. As the set of parameters used are specific to the problem at hand, this metric needs also to be problem-specific. Several optimization algorithms can be used for this; in fact, in a previous work, Ozik et al. (2019) used a genetic algorithm to find optimal points that produced simulations that had counts with the lowest final mean tumor cell as well as an active learning algorithm to build surrogate models for characterizing the parameter space structure based on different viability thresholds.

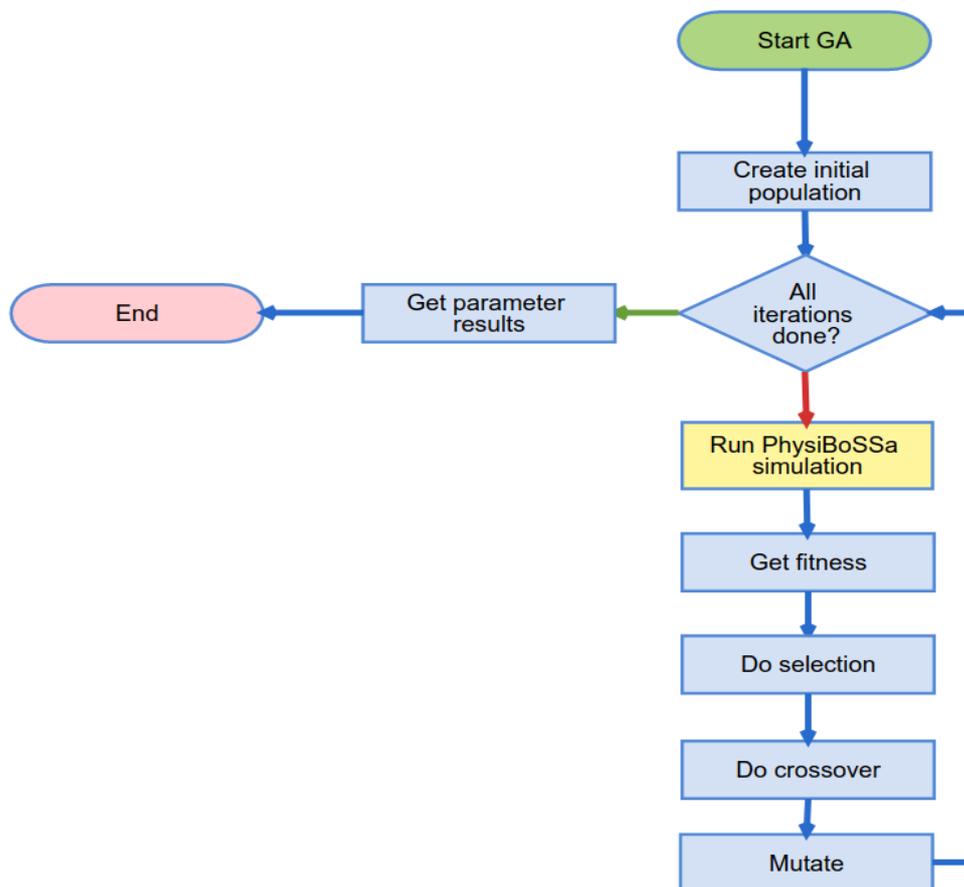
In our case, we have used a genetic algorithm and a random forest strategy to find a proper set of values for the parameters in PhysiBoSSa (Table 10) that allow for the reproduction of the results from the simulation with a pulsating regime of TNF each 150 minutes. In this section we detail some characteristics of these algorithms that have proved useful for our goals, but more in-depth information on the use of these algorithms can be found in Deliverable 6.2.

 Project supported by the European Commission Contract no. 825070	<h3>WP1 T1.1-T1.3</h3> <h2>Deliverable D1.2</h2>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

The genetic algorithm from python's DEAP library (<https://github.com/DEAP/deap>) was used with two sets of settings (Table 11) and following a classic GA strategy (Figure 17).

**Table 11: Sets of parameters used in the Genetic Algorithm studies.**

	BSC set	NCSR set
Crossover probability	0.5	0.7
Mutation probability	0.2	0.5
Individuals' characteristics	7 parameters	4 parameters
Population	150	20
Number of generations	30	20



**Figure 17: Flowchart of the genetic algorithm used on PhysiBoSSa.**

Additionally, we used a random forest algorithm from python's scikit-learn library (Pedregosa *et al.*, 2011) to study the structure of the parameter space. While genetic algorithms can be very efficient in discovering optimal solutions in large spaces, they are not sufficient for estimating the structure of complex parameter spaces, a task where random forests can provide useful insight.

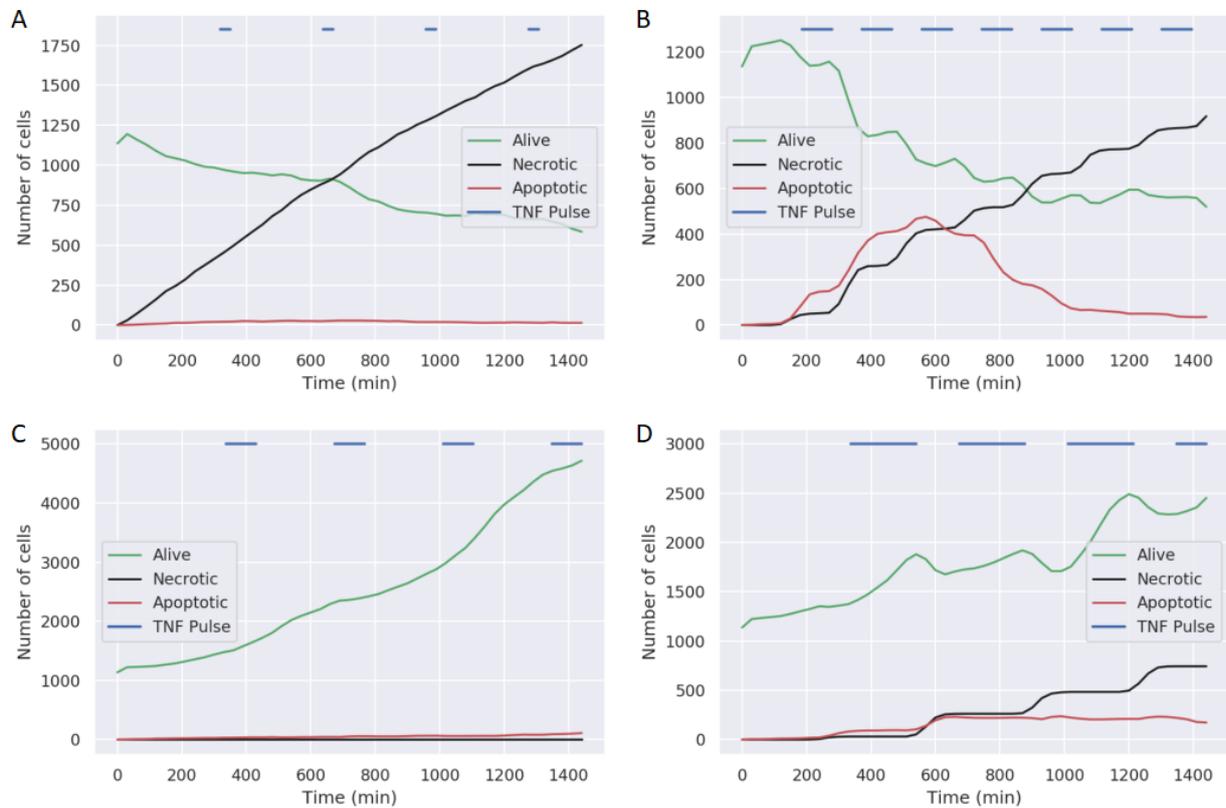
The training dataset used is a set of simulations classified via 1-NN and known cases as interesting or non-interesting and constitutes the initial random forest seed. From here, the algorithm iterates over a set of steps in which it constructs

 Project supported by the European Commission Contract no. 825070	<h2>WP1 T1.1-T1.3</h2> <h1>Deliverable D1.2</h1>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public

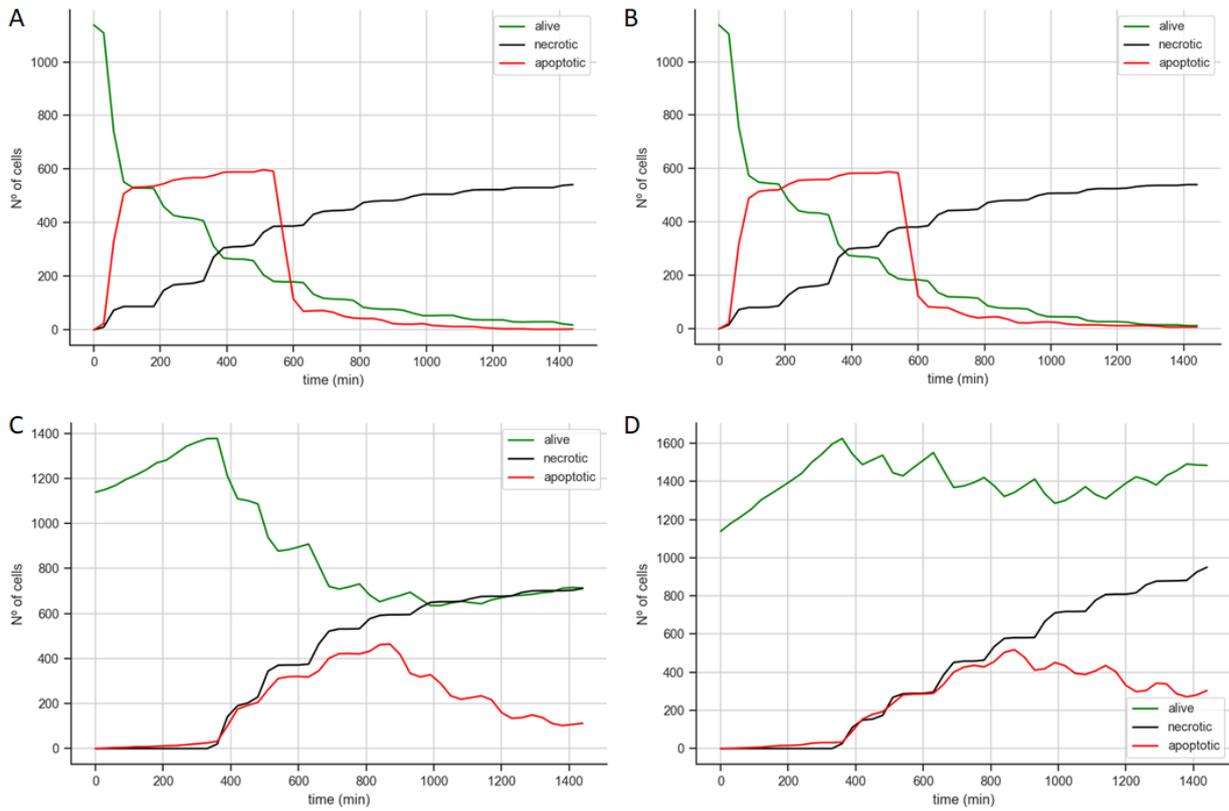
a random forest; it creates a fine grid in the parameter space and classifies its points using the new random forest; then computes the class uncertainty and keeps the highest uncertainty values, as candidate points; and if the number of candidate points is bigger than a threshold ( $k$ ), it clusters the candidate points into  $k$  groups. Finally, the algorithm selects one point from each cluster at random, evaluates the selected points by running a PhysiBoSS simulation and using a 1-NN classifier and adds these points to the dataset.

## 6.3 Preliminary results

These ongoing efforts allow us to define the structure and hierarchy of the model's parameters and to evaluate its sensibility to the parameters' perturbation and are necessary to ensure the adaptability of present multiscale modeling to other models and use cases. We have found sets of data that capture the desired behaviors of our model as well as sets of data that fail at capturing these behaviors. We hereby present some of the preliminary results (Figure 18 and Figure 19) using the aforementioned algorithms to find ranges of parameters that cause the model to behave like in Figure 16.



**Figure 18: Some representative simulations performed by the model exploration framework on NCSR parameter set. (A and B) Positive examples of parameter sets that tally the behavior of Figure 16 (C and D) Negative examples of parameter sets that do not tally the behavior of Figure 16.**



**Figure 19: Some representative simulations performed by the model exploration framework on BSC parameter set. (A and B) Positive examples of parameter sets that tally the behavior of Figure 16 (C and D) Negative examples of parameter sets that do not tally the behavior of Figure 16.**

 <p>Project supported by the European Commission Contract no. 825070</p>	<h2>WP1 T1.1-T1.3</h2> <h3>Deliverable D1.2</h3>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



## 7 Conclusions and Future Work

In this deliverable, we have described in detail the agent-based model initially used in the INFORÉ Life Science use case. First, we have introduced the different uses of agent-based modeling in biomedical projects and explained the reasons behind our choice of center-based agents for this project as the basis of the multiscale model (MSM).

Second, we have detailed how this agent-based model is integrated into the MSMs, the different modules of this MSM, their parameters and how the different modules are connected and build up a MSM. These modules consist of:

- the environment component that simulates all the diffusion, creation and uptake of chemical entities that roam in the environment;
- the agent-based component that takes care of the population level and simulates the cells dynamics, their growth, death, movement and overall physical behavior among cells and among them and their surrounding environment;
- the signaling network module that takes care of the individual cells' level and simulates the behavior of each cell in response to its environment and its neighboring conditions; and
- the cell cycle module that takes care of how the cells grow and divide.

We still have in scope an additional metabolic module whose integration is an ongoing work that would take care of the energy and biomass fluxes of the cells.

Third, we have detailed the ongoing implementation of an HPC version of our MSM that will allow its scaling-up to clusters and enable INFORÉ to reach its KPIs. We have started by parallelizing the fastest of all the time scales considered in our MSM, the environment one and its Thomas solver used to simulate the diffusion of all chemical entities by using domain partitioning. We present benchmarks on different environmental domains that show a speed-up in the simulations and, more importantly, that the parallelization is possible. Currently, we are parallelizing the agents' component using the same domain partition strategy with promising results.

Four, we have presented a model exploration framework that we are using to explore the parameter space of our pulsating TNF simulations. The goal is to define the structure and hierarchy of the model's parameters and to evaluate its sensibility to the parameters' perturbations. As expected, this exploration retrieves sets of parameters that reproduce the desired behavior and others that do not. Next, we plan to use this model exploration framework with different optimization algorithms to expand the TNF experiments to other types of drug regimens and to study the effect of changes in drug availability combined with microenvironment set-ups in 3D spheroids in a gastric adenocarcinoma cell-line multiscale model.

 Horizon 2020 European Union Funding for Research & Innovation	Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
			Rev.:	1.0
			Date:	30/04/2020
			Class.:	Public

## 8 References

- An,G. (2010) Closing the Scientific Loop: Bridging Correlation and Causality in the Petaflop Age. *Sci. Transl. Med.*, **2**, 41ps34-41ps34.
- Anderson,A.R.A. (2005) A hybrid mathematical model of solid tumour invasion: The importance of cell adhesion. *Math. Med. Biol.*, **22**, 163–186.
- Anderson,A.R.A. *et al.* (2006) Tumor Morphology and Phenotypic Evolution Driven by Selective Pressure from the Microenvironment. *Cell*, **127**, 905–915.
- Armstrong,T.G. *et al.* (2014) Compiler Techniques for Massively Scalable Implicit Task Parallelism. In, *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, New Orleans, LA, USA, pp. 299–310.
- Béal,J. *et al.* (2019) Personalization of logical models with multi-omics data allows clinical stratification of patients. *Front. Physiol.*, **9**, 1965.
- Bernabeu,M.O. *et al.* (2010) Shock-induced arrhythmogenesis in the human heart: A computational modelling study. *Conf. Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, **2010**, 760–763.
- Calzone,L. *et al.* (2018) Logical versus kinetic modeling of biological networks: applications in cancer research. *Curr. Opin. Chem. Eng.*, **21**, 22–31.
- Chandra,R. *et al.* (2001) Parallel programming in OpenMP Morgan Kaufmann Publishers, San Francisco, CA.
- Chapman,B. *et al.* (2008) Using OpenMP: portable shared memory parallel programming MIT Press, Cambridge, Mass.
- Cohen,D.P.A. *et al.* (2015) Mathematical Modelling of Molecular Pathways Enabling Tumour Cell Invasion and Migration. *PLoS Comput Biol*, **11**, e1004571.
- Flobak,Å. *et al.* (2015) Discovery of Drug Synergies in Gastric Cancer Cells Predicted by Logical Modeling. *PLOS Comput. Biol.*, **11**, e1004426.
- Gérard,C. and Goldbeter,A. (2011) A skeleton model for the network of cyclin-dependent kinases driving the mammalian cell cycle. *Interface Focus*, **1**, 24–35.
- Ghaffarizadeh,A. *et al.* (2016) BioFVM: an efficient, parallelized diffusive transport solver for 3-D biological simulations. *Bioinformatics*, **32**, 1256–1258.
- Ghaffarizadeh,A. *et al.* (2018) PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. *PLOS Comput. Biol.*, **14**, e1005991.
- Graner,F. and Glazier,J.A. (1992) Simulation of biological cell sorting using a two-dimensional extended Potts model. *Phys. Rev. Lett.*, **69**, 2013.
- Grimm,V. *et al.* (2006) A standard protocol for describing individual-based and agent-based models. *Ecol. Model.*, **198**, 115–126.
- Hoehme,S. *et al.* (2010) Prediction and validation of cell alignment along microvessels as order principle to restore tissue architecture in liver regeneration. *Proc. Natl. Acad. Sci.*, **107**, 10371–10376.
- Hoehme,S. and Drasdo,D. (2010) A cell-based simulation software for multi-cellular systems. *Bioinformatics*, **26**, 2641–2642.
- Kamil,S. *et al.* (2010) An auto-tuning framework for parallel multicore stencil computations. In, *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pp. 1–12.
- László,E. (2016) Parallelization of numerical methods on parallel processor architectures.
- Letort,G. *et al.* (2018) PhysiBoSS: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. *Bioinformatics*, bty766.
- Macal,C.M. and North,M.J. (2010) Tutorial on agent-based modelling and simulation. *J. Simul.*, **4**, 151–162.
- Message Passing Interface Forum (2015) MPI: A message-passing interface standard version 3.1.
- Mirams,G.R. *et al.* (2013) Chaste: An Open Source C++ Library for Computational Physiology and Biology. *PLOS Comput. Biol.*, **9**, e1002970.
- Montagud,A. *et al.* (2017) Conceptual and computational framework for logical modelling of biological networks deregulated in diseases. *Brief. Bioinform.*, bbx163.
- OpenMP Architecture Review Board (2018) OpenMP application program interface version 5.0.
- Osborne,J.M. *et al.* (2017) Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLOS Comput. Biol.*, **13**, e1005387.
- Ozik,J. *et al.* (2016) From desktop to large-scale model exploration with SWIFT/T. *Proc. Winter Simul. Conf. Winter Simul. Conf.*, **2016**, 206–220.
- Ozik,J. *et al.* (2018) High-throughput cancer hypothesis testing with an integrated PhysiCell-EMEWS workflow. *BMC Bioinformatics*, **19**, 483.

 Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
		Rev.:	1.0
		Date:	30/04/2020
		Class.:	Public



- Ozik, J. *et al.* (2019) Learning-accelerated discovery of immune-tumour interactions. *Mol. Syst. Des. Eng.*
- Pas, R. van der *et al.* (2017) Using OpenMP--the next step: affinity, accelerators, tasking, and SIMD The MIT Press, Cambridge, Massachusetts.
- Pedregosa, F. *et al.* (2011) Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
- Rabenseifner, R. *et al.* (2009) Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In, *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing.*, pp. 427–436.
- Saxena, G. *et al.* (2016) A cache-aware approach to domain decomposition for stencil-based codes. In, *2016 International Conference on High Performance Computing Simulation (HPCS).*, pp. 875–885.
- Saxena, G. *et al.* (2018) A quasi-cache-aware model for optimal domain partitioning in parallel geometric multigrid. *Concurr. Comput. Pract. Exp.*, **30**, e4328.
- Saxena, G. (2018) Efficient Domain Partitioning for Stencil-based Parallel Operators.
- Shan, M. *et al.* (2018) Multi-scale computational study of the Warburg effect, reverse Warburg effect and glutamine addiction in solid tumors. *PLOS Comput. Biol.*, **14**, e1006584.
- Smith, L. and Bull, M. (2001) Development of Mixed Mode MPI / OpenMP Applications. *Sci. Program.*, **9**, 83–98.
- Stoll, G. *et al.* (2012) Continuous time Boolean modeling for biological signaling: application of Gillespie algorithm. *BMC Syst. Biol.*, **6**, 116.
- Stoll, G. *et al.* (2017) MaBoSS 2.0: an environment for stochastic Boolean modeling. *Bioinformatics*, **33**, 2226–2228.
- Swat, M.H. *et al.* (2012) Multi-Scale Modeling of Tissues Using CompuCell3D. In, *Methods in Cell Biology*. Elsevier, pp. 325–366.
- Thomas, L. (1949) Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory Columbia University, New York, NY.
- Trisilowati and Mallet, D.G. (2012) *In Silico* Experimental Modeling of Cancer Treatment. *ISRN Oncol.*, **2012**, 1–8.
- Tyson, J.J. *et al.* (2003) Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr. Opin. Cell Biol.*, **15**, 221–231.
- Van Liedekerke, P. *et al.* (2019) Quantitative cell-based model predicts mechanical stress response of growing tumor spheroids over various growth conditions and cell lines. *PLOS Comput. Biol.*, **15**, e1006273.
- Yang, R. *et al.* (2018) Cell type-dependent bimodal p53 activation engenders a dynamic mechanism of chemoresistance. *Sci. Adv.*, **4**, eaat5077.

 Horizon 2020 European Union Funding for Research & Innovation	Project supported by the European Commission Contract no. 825070	<b>WP1 T1.1-T1.3</b> <b>Deliverable D1.2</b>	Doc.nr.:	WP1 D1.2
			Rev.:	1.0
			Date:	30/04/2020
			Class.:	Public

## 9 Glossary

- **AGS:** name of a cell line derived from gastric adenocarcinoma used as a laboratory model to study the aforementioned disease ([https://web.expasy.org/cellosaurus/CVCL\\_0139](https://web.expasy.org/cellosaurus/CVCL_0139)).
- **Apoptosis:** programmed cell death that occurs in multicellular organisms.
- **Cell cycle:** cell-division cycle, is the series of events that take place in a cell leading to duplication of its DNA and division of its cytoplasm and organelles to produce two daughter cells<sup>2</sup>.
- **Drug synergy:** positive interaction between two or more drugs that causes the total effect of the drugs to be greater than the sum of the individual effects of each drug.
- **Necrosis:** traumatic cell death caused by acute cellular injury or lack of nutrients.
- **Resistant cell:** cell that becomes immune to a specific drug or treatment.
- **Signaling pathway:** Is a set of proteins in a cell that work together to translate external signals and to control one or more cell functions, such as cell division or cell death<sup>3</sup>. The emergence of cancer cells is closely related to the deregulation of malfunctioning of specific signaling pathways.

<sup>2</sup> [https://en.wikipedia.org/wiki/Cell\\_cycle](https://en.wikipedia.org/wiki/Cell_cycle)

<sup>3</sup> [https://en.wikipedia.org/wiki/Cell\\_signaling#Signaling\\_pathways](https://en.wikipedia.org/wiki/Cell_signaling#Signaling_pathways)

 <p>Project supported by the European Commission Contract no. 825070</p>	<h3>WP1 T1.1-T1.3</h3> <h2>Deliverable D1.2</h2>	<b>Doc.nr.:</b>	WP1 D1.2
		<b>Rev.:</b>	1.0
		<b>Date:</b>	30/04/2020
		<b>Class.:</b>	Public