



Formal description of Complex Automata, including report on execution models, and draft version of complex automaton model for prototype application

Deliverable 2.2 (M12-M24)

Project Start:	01/09/2006
Project Duration:	36 Months
Priority area	2.3.4. (xi)
Contract No.:	INFSO-IST-033446
Website:	http://www.complex-automata.org

Due-Date:	month 24
Delivery:	month 24
Lead Partner:	<i>UNIGE</i>
Dissemination Level:	<i>PU</i>
Status:	<i>Final</i>
Approved:	WP2
Version:	2.0

Log of Document

Version	Date	Changes Summary	Authors
0.9	13.08.08	Setting up the document structure	B. Chopard UNIGE
	13.08.08	Executive summary + intro	B. Chopard
	14.08.08	Chapitre 2 (2.5 to be completed)	B. Chopard
	14.08.08	Chapter 3	A. Caiazza
	18.08.08	Execution Model	JL Falcone
	18.08.08	Drug Diffusion	JL Falcone
	18.08.08	SMC	Bernd Stahl
	18.08.08	Text integration	JLF+BC
1.0	19.08.08	Appendix, Reference	BC+JLF
1.1	20.08.08	Final editing and formatting, making ready for internal review	A. Hoekstra, UvA
1.2	22.08.08	Corrections according to AGH review + some improvement	JLF + BC, UNIGE
1.3	23.08.08	Corrections and comments	PVL
1.4	26.08.08	Validation of PVL correction	JLF+BC
2.0	26.08.08	Final Version	BC+JLF

Table of Contents

Table of Contents.....	3
List of Acronyms	3
Executive Summary	5
Introduction	6
Section 1: Complex Automata Approach: A draft Story.....	7
1.1 Introduction	7
1.2 Classification and taxonomy	8
1.3 Coupling template	10
1.4 Coupling strategies	11
1.5 Execution model	12
1.6 Scale-splitting error	18
1.7 Summary of the CxA methodology	22
Section 2: The In-stent restenosis as a CxA.....	23
2.1 Towards the computational model	24
2.2 Multiscale Problem	24
2.3 Computational description of single scale models	28
2.4 Scale Separation Map	38
Section 4: References.....	39
5.1 Coupling Templates	42
5.2 Geometrical details of the model	46

List of Acronyms

ABM	Agent Base Model
BF	Bulk Flow
CA	Cellular Automata
CxA	Complex Automata
DoW	Description of Work (Coast Document)
DD	Drug Diffusion
EC	Endothelial Cells
FE	Finite Element Model
IC	Initial Conditions
IEL	Internal Elastic Lamina
ISR	In-stent restenosis
LBM	Lattice Boltzmann Models
MD	Molecular Dynamics Model
mD	multi-Domain
MM	Multiscale and Multiscience
RD	Reaction-Diffusion

RTF	Rich Thrombus Formation
sD	single-Domain
SEL	Submodel Execution Loop
SMC	Smooth Muscle Cells
SSM	Scale Separation Map
TL	Transport Layer
UML	Unified Modeling Language

Executive Summary

Deliverable D2.2 reports on the activities as carried out in Workpackage 2, tasks 2.4 – 2.6, during Y2 of the project. Task 2.4 (Execution model) describes how to interoperate the various single-scale submodels that form a given multiscale model. Task 2.5 (Integration) aims to consolidate the foundation of the Complex Automata (CxA) formalism. Task 2.6 (prototype application) defines how the target application of the COAST project can be expressed in the CxA approach.

The present document first presents the current state of the CxA theory. Some scale separation strategies are proposed and expressed in terms of the CxA quantities. The execution model is then described and we show that, conceptually, the CxA formalism is well adapted to the coupling of existing sub-models through an asynchronous data-driven model. Finally a mathematical treatment of the scale separation process (i.e. the splitting of a two-scale model into two single-scale models) is derived in great detail for a generic problem: reaction-diffusion processes. This analysis gives a quantitative estimate of the error resulting from the scale splitting process.

The prototype application concerns the modeling of restenosis in a stented coronary artery. This document describes the main processes that are involved in this problem. Following the CxA methodology, they are first organized on the scale separation map (SSM). The role of each process is then discussed and their mutual coupling, in terms of the CxA quantities, is proposed.

Introduction

The objective of Workpackage 2, Complex Automata, is “to realize a mathematical foundation of the concept of Complex Automata (CxA), and to identify the main mechanisms of coupling automata spanning length and time scales, leading to a formal modeling language for Complex Automata” (quotation from the Description of Work). This deliverable reports on the work carried out in Workpackage 2 during the second year of the project.

This document first presents a draft of the CxA theory. It reflects the activities of task 2.4 and 2.5, as well as several papers that have been written during Y2 of the project. These papers have been presented (or will be presented soon) to international conferences (ICCS 2008, ACRI 2008, for details we refer to the Periodic Activity report for Y2 of the project, deliverable D1.5).

Compared to the Y1 deliverable D2.1, this document proposes a more integrated version of the formalism, with a detailed description of the execution model and a mathematical analysis of coupling strategies. The concepts already described in D2.1 are only briefly repeated here for the consistency of the discussion.

In a second part we discuss progress in task 2.6 concerning the definition of the problem of “In-stent restenosis” (ISR) within the tools offered by the CxA model.

Note that in task 2.5 (Integration) some work has been done to apply the CxA approach to applications outside the COAST scope. Specifically this is the case of a transport and aggregation process in volcanic eruption. This activity will be discussed in detail in the Y3 deliverable and will also be reported in the context of task 4.6 in WP4.

As outlined in the COAST DoW, the final version of the theory will be finalized in month 36. We plan to include more mathematical derivations describing the benefit (in term of computational speedups) and the disadvantages (in terms of loss of accuracy) of other coupling strategies. We will also show how the CxA framework can provide the building blocks of a multiscale modeling language.

Section 1: Complex Automata Approach: A draft Story

1.1 Introduction

Cellular Automata (CA), Lattice Boltzmann models (LBM) and Agent Based Models (ABM) are generally acknowledged to be a powerful way to describe and model complex natural phenomena [Chopard 1998, Deutsch 2005, Slood 2007].

The ever-increasing availability of experimental data on every scale, from 'atom to material' or from 'gene to health', in combination with the similar ever-increasing computational power [Bader08, Hoekstra 2008], facilitates the modeling and the simulation of natural phenomena, taking into account all required spatial and temporal scales, see e.g. [Slood 2007b].

Multiscale modeling and simulation, as a paradigm in Computational Science, is becoming more and more important, as witnessed by e.g. dedicated special issue [IEEE 2005] and thematic journals [SIAM, IJMCE].

Our main goal in the COAST project is to use CA and related methods to model these multi-scale processes efficient simulations on digital computers.

When using CA to model a system we denote by $A(\Delta x, \Delta t, L, T)$ its spatio-temporal domain, which is made of cells of size Δx and spans a region of size L , while the quantity Δt is the time step and T is the end of the simulated time interval. Therefore, processes with time scales between Δt and T can be represented and spatial scales ranging from Δx to L can be resolved. When executing such CA on a digital computer we the execution time T_{ex} is proportional to

$$T_{ex} \sim \frac{T}{\Delta t} \left(\frac{L}{\Delta x} \right)^D \quad (1)$$

where D is the spatial dimension of the simulated domain. Trying to model a multiscale system with a single CA requires choosing Δx and Δt in such a way that the smallest microscopic details and fastest dynamic response of the system are captured, yet the overall system size L and slowest dynamic time scale T need to be covered. For instance, in modeling human physiology the relevant range of spatial scales is from nanometers to meters (i.e. a factor 10^9) whereas the

temporal scale is from microseconds to the normal human lifespan (i.e a factor 10^{15}). Such numbers, in combination with Eq. (1) immediately show that one will never be able to simulate multiscale systems with a single CA spanning such a wide range of scales.

For this reason we introduced the concept of **Complex Automata** (CxA) as a set of single-scale CA's, LBM and/or ABM (i.e processes taking place at specific spatio-temporal scales) coupled together so as to approximate the original, full, multiscale problem. This idea is illustrated in Fig.1, using the concept of the Scale Separation Map (SSM). On the SSM each spatio-temporal system is represented as a rectangle the lower-left and upper-right corners of which indicate the smallest and the largest scale that can be described. The SSM turns out to be a very powerful way to describe a multiscale, multiscale problem. It is therefore a central element of the CxA methodology and the starting point of the problem specification.

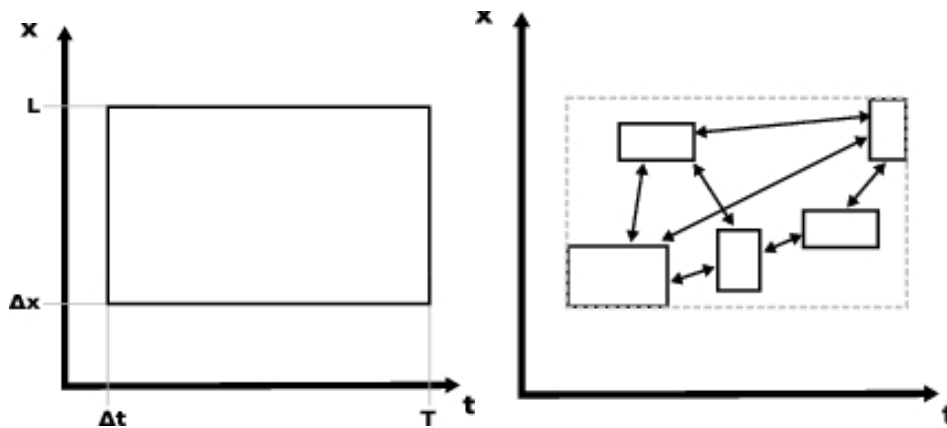


Fig. 1: From a single multiscale model to many single-scale models. The horizontal axis of the SSM represents the temporal scales and the vertical axis the spatial scales.

1.2 Classification and taxonomy

In this section we briefly review how the CxA formalism yields a new taxonomy for multiscale, multiscale (MM) problems. We refer the reader to deliverable D2.1 for more detail.

Based on splitting a MM problem on the SSM, several relationships between the different single-scale processes (or submodels) can be observed. In addition, considering whether or not two submodels share the same computational domain, we can formulate a classification of multiscale problems. We use the term multi-Domain (mD) to describe a situation where the computational domains are distinct or partially overlap. Single-Domain (sD) refers to problems where all domains are the same.

Our classification is illustrated in Fig.2 for the case of a problem that splits into two submodels. Of course, the proposed classification scheme and taxonomy can be extended straightforwardly to more than two single-scale processes. Note also the important fact that the present formalism not only describes a micro-macro coupling as is the case in most published work, but also several new coupling relationships that have been overlooked in previous multiscale approaches.






	Time Overlap		Time Separation	
Space Overlap	Single Domain Coupling through collision operator. <i>Snow transport, diffusion/ advection, ...</i>	 Multi Domain Coupling through boundary condition. <i>Fluid structure, grid refinement, ...</i>	Single Domain Coupling through collision operator. <i>Forest-Savannah-Fire interactions</i>	 Multi Domain Coupling through boundary, initial conditions. <i>Coral Growth, ...</i>
	Single Domain Coupling through collision operator. <i>Algae-Water ecological model, ...</i>	 Multi Domain Coupling through boundary condition. <i>Wave propagation in two media, ...</i>	 Hierarchical Coupling Coupling through collision operator and initialization. <i>Suspension Fluid, ...</i>	
Space Separation			 "Physics-Biology Coupling" Coupling through boundary conditions and initialization. <i>Oscillating blood flow and endothelial cells, ...</i>	

Fig.2 Taxonomy and classification scheme resulting from the CxA methodology. Some examples of applications are given to illustrate the classification scheme.

1.3 Coupling template

Formally, a Cellular Automaton can be defined as

$$CA = \{A(\Delta x, L, \Delta t, T), \mathbb{F}, \Phi, f_{init} \in \mathbb{F}, \mathbf{u}\} \quad (2)$$

specifying the spatio-temporal domain A , with discretization parameters $(\Delta x, \Delta t)$, the space of states \mathbb{F} , the initial condition f_{init} and the update rule Φ . In Eq. (2) we include in the definition a field \mathbf{u} collecting the external data on which the CA depends.

The state of the system is described by $\hat{f}^{tn} \in \mathbb{F}$, denoting the numerical solution at the n -th time step:

$$\begin{aligned} \hat{f}^0 &= f_{init}[u_0], \text{ initial condition} \\ \hat{f}^{t_{n+\Delta t}} &= \Phi[\mathbf{u}; \hat{f}^n], \quad n > 0 \end{aligned} \quad (3)$$

with u_0 an external field connected to the initial condition. Additionally, restricting to CA, LBM or ABM, we constrain the update rule Φ to the form

$$\Phi[\mathbf{u}; f] = (B[u_B] \circ P \circ C[u_C])[f], \quad (4)$$

i.e. written as a combination of three operators: **collision** $C[u_C]$ dependant on external parameters u_C , **propagation** P , dependant on the topology of the domain, and **boundary** $B[u_B]$, dependant on external parameters denoted by u_B . This form of a CA update rule is inspired by Lattice Gas Automata (see e.g. [Chopard 1998, Slood 2007, Chopard 2008]).

As the dynamics of all submodels of a CxA can be defined with the same sequence of operators, the so-called submodel Execution Loop (SEL) is the second central ingredient of our multiscale framework. It allows us to define couplings between submodels through a finite number of generic *coupling templates*, as illustrated in Fig. 3 for a specific problem (coupling of a fine and coarse computational grid). The arrows indicate a flow of information from one system to the other. Their starting points are the operator of the SEL which produces the data and the end points indicate which operator utilises it.

A third important ingredient of the CxA formalism is that the coupling (i.e. the arrows in Fig 3) is realized through *smart conduits* which are

software components facilitating data transfer and the possible conversion of data-structure between the source and the destination.

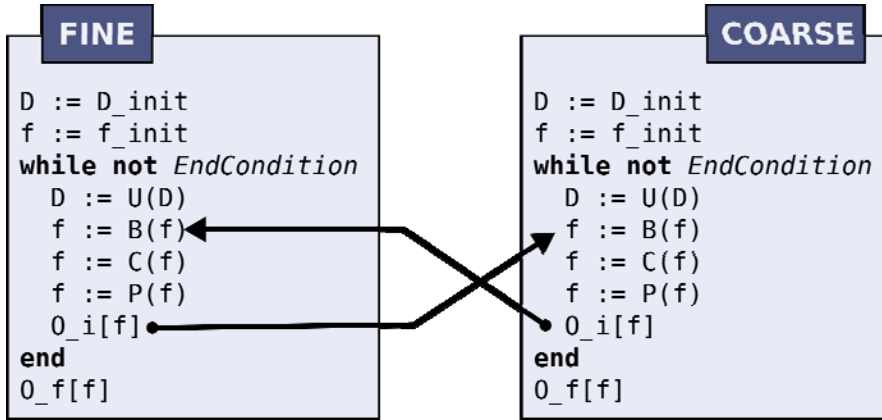


Fig. 3: Illustration of the SEL's and their coupling in a LB multiblock grid refinement. In addition to the P , C , and B operators already described, observables operators O are also introduced. In this example, the values of the f 's in one system is used to feed the Boundary operator of the other.

1.4 Coupling strategies

Although not really part of the COAST DoW, it is interesting to identify and express several coupling/splitting strategies using the CxA framework, in particular the collision and propagation operators C and P . For the sake of discussion, let us add the spatial scale Δx as a subscript to P and C . We also assume that the time scale Δt is related to Δx by some simple function (for instance $\Delta x \sim \Delta t^2$ or $\Delta x \sim \Delta t$).

- 1. Scale splitting:** This coupling strategy may apply for a single domain (sD) problem, when the full-scale model execution loop reads

$$P_{\Delta x} C_{\Delta x} = P_{\Delta x} C_{\Delta x}^{(1)} C_{\Delta x}^{(2)} \quad (5)$$

Then, if $C^{(1)}$ and $C^{(2)}$ act at different scales we may write

$$[P_{\Delta x} C_{\Delta x}]^M = P_{M\Delta x} C_{M\Delta x}^{(1)} [C_{\Delta x}^{(2)}]^M \quad (6)$$

if M is not too large. This strategy means that one performs M steps of the fast process $C^{(2)}$ for one step of the slow process $C^{(1)}$, but on a scale M times bigger. A detailed example of this approach is given in section 2.6 for a reaction-diffusion problem.

2. Coarse graining: This strategy consists for instance in writing a single domain (sD) problem as

$$[P_{\Delta x}C_{\Delta x}]^n = \Gamma^{-1}[P_{2\Delta x}C_{2\Delta x}]^{n/2}\Gamma \quad (7)$$

where Γ is some transformation matrix. The above decomposition assumes that two steps of the original model can be approximated by one step of renormalized collision and propagation operators, acting on a two times larger scale. This approach corresponds to the grid refinement problem. The validity of Eq. (7) for LB grid refinement is currently under investigation.

3. Amplification: In many cases, one process acts with low intensity but for many time steps in an environment which changes periodically with time. For instance a pulsatile flow induces some growth phenomena on the walls of a tube. We assume we have a multi-domain problem with two submodels iterated for n steps.

$$[P^{(1)}C^{(1)}]^n \quad \text{and} \quad [P^{(2)}C^{(2)}(k)]^n \quad (8)$$

where k is a parameter measuring the intensity of process 2. If we assume that the period of this process 1 is $m \ll n$. Then, we can write the above dynamics as

$$[P^{(1)}C^{(1)}]^m \quad \text{and} \quad [P^{(2)}C^{(2)}(k')]^m \quad (9)$$

Where $k' > k$ is a new intensity of process 2. For a linear collision operator we would have $k = (n/m)k$.

1.5 Execution model

Coupling several submodels, using coupling templates and smart conduits raises implementation issues. A typical situation is shown in Fig. 3. Below we explain the main concepts of the proposed execution model. The proposed model is compatible with the asynchronous channel actor-model framework [Agha 1986]. Note that details of the technical implementation are described in WP3 deliverables.

1.5.1 CxA Components

CxA are directed bipartite graphs whose edges represent a single direction communication channel and the vertices are either **kernels** or **conduits**. These components have the following definitions:

- The **kernels** are the main computational units of the CxA.

Generally, kernels are the single-scale submodel solvers as described above. However, when needed, they can also execute other tasks such as; measurements, complex mappings, etc.

- The **conduits** are “smart” communication channels, related to unix named pipes. Each conduit connects a pair of kernels together in an oriented fashion and, in principle, only one physical quantity is transported per conduit. These conduits are composed of three parts: (1) an incoming buffer, the **entrance**; (2) an outgoing buffer, the **exit**; and eventually, (3) one or several data **filters** (for interpolation, rescaling, restriction, discretization, etc.) Conduits works in a purely reactive way: when data is copied at the entrance, the conduits apply the filters and move the data into the outgoing buffer.

Each conduit is connected to only two kernels, but kernels can be connected to an arbitrary number of conduits. Each component is either a full process or a thread depending on the implementation. They can reside in the same machine or be distributed across the network.

1.5.2 CxA Communications

In CxA, kernels communicate exclusively via conduits, using a message passing paradigm. Only two communication primitives are defined to interact with conduits:

1. `send(data)`: this primitive sends a data vector from a kernel to a conduit entrance. It is **non-blocking**, since it returns as soon as the data is sent to the conduit, whether or not the destination process has read the data or not. This corresponds to a push communication.
2. `receive()`: this primitive allows a kernel to receive data from a conduit exit. This primitive is **blocking**, it will return only when the desired data exist in the conduit. The receiving kernel will then simply wait until the data is available before resuming its computations. This corresponds to a pull communication.

Conduits entrances and exits are supposed to have large buffers, able to store several large data structures. These buffers act as FIFO where each entry is a reference to a data-structure. So, if the sending kernel is faster than the receiving one, several data vectors will be stored in the exit buffer, waiting for a `receive()` call from the destination kernel. The FIFO nature of the buffer ensures that the data are always read in the correct time order.

The actual communication can be either a memory copy if the kernel and conduit reside in the same processor, or a network communication if both components reside on different machines.

For instance, the coral model SEL represented in Fig. 4, can be rewritten as follows, to include explicitly the two communication primitives:

```
while not EndCondition
    D := U(D)
    DomainConduit.send( D )
    f := B(f)
    velocityMap := VelocityConduit.receive()
    f := C(f,velocityMap)
    f := P(f)
end
```

1.5.3 CxA Initialization and start

CxA initialization occurs in a semi-decentralized way. First, each conduit and kernel is spawned (possibly on several machines). Then a special process, termed **plumber**, is responsible for connecting each kernel with the entrances and exits of the relevant conduits. The plumber terminates as soon as this basic task is finished. The rest of the initialization process is then fully decentralized:

1. As soon as a **kernel** is fully connected with the required conduits exits and entrances, it starts its computations. If it is sending data to a yet unconnected kernel, the data will be kept in the conduit until the receiver is active and reading. On the other hand, if a conduit tries to receive data originating from an unconnected kernel, it will hang until the sending kernel connects and transmits data.
2. For **conduits** the situation is even simpler. Since they are purely reactive components, nothing will happen in an unconnected conduit. Similarly, if only the conduit exit is connected, the conduit will do nothing. In contrast, if only the conduit entrance is connected, the conduit will simply process incoming data which will be accumulated in the exit buffer. Therefore, the conduit is always in a valid state (assuming it has enough internal memory).

1.5.4 CxA Synchronization

CxA graphs are usually cyclic. Even the basic examples with just two single-scale models (see Fig. 4) will display a communication cycle if both models can influence one another. Moreover CxA are multiscale systems and kernels can thus function at different time scales. These two properties make a central scheduler approach impractical.

However, the fact that the receive primitive is blocking and the send is non-blocking, allows a data-driven synchronization to occur naturally. Indeed, kernels will just wait until information is available before continuing their computation. An example of such synchronization is shown in Fig. 5 for the coral model.

The main problem with this method are possible deadlock situations. However, such issues can be easily prevented with the CxA execution model. In the coral example, deadlock is avoided by having a model (the coral) which sends before receiving. This allows the flow model to continue its computations to produce the data that will unlock the coral, etc. In contrast, the situation presented in Fig. 3 will produce a deadlock because both models try to receive before sending anything. This problem is easily solved by adding send instructions before entering the submodel execution loop.

Furthermore, the fact that communication is pairwise and that the conduits use buffers, makes the race condition impossible. Data are meant to be read by only one process, data sent in a conduit entrance will be processed only by that conduit and data moved to conduit exits will concern only a single kernel.

1.5.5 CxA Termination

The termination of the whole CxA is also designed to be fully decentralized:

- When a kernel finishes its computations (because of a maximal time, a steady state condition, etc.), it first notifies all its conduits and then it terminates itself.
- Similarly, when a conduit receives termination notifications from all connected kernels, it can terminate itself.

While the conduit termination rule is always safe (a conduit stops when no kernel is connected anymore), the kernel termination rule needs a extra mechanism. Otherwise, a problem occurs if a kernel is waiting for information from an already terminated kernel. For instance, in the coral example (Fig. 4) the flow model will hang on for the domain update, even after the coral model termination.

This issue is solved by introducing a **stop signal** able to release a kernel blocked in the receive primitive. This signal is propagated by a kernel through the existing conduits, using a third primitive:

- `stop()`: this primitive sends the stop signal through a conduit.

The receive primitive is then modified slightly. It works exactly as seen above but can return either the expected data or the stop signal.

Therefore a kernel waiting for data can be released by a stop signal. Kernels are then responsible to send, treat and propagate stop signals. Generally a kernel receiving a stop signal should:

1. Abort the submodel execution loop.
2. Send some final data, if required.
3. Propagate the stop signal to each connected conduit entrances.
4. Notify each connected entrance and exit.
5. Terminate itself gracefully.

With this termination scheme, all kernels which need data from the rest of the CxA will thus stop. The stop signal can originate from any kernel, and this approach also works if two (or more) kernels reach a stop condition at the same time.

For instance, we can add a stop mechanism to the example of Fig. 4, as follows:

1. Coral:

```
while not EndCondition
    D := U(D)
    DomainConduit.send( D )
    f := B(f)
    velocityMap := VelocityConduit.receive()
    f := C(f,velocityMap)
    f := P(f)
end
DomainConduit.stop()
myStop()
```

2. Flow:

```
While true
    domain := DomainConduit.receive()
    if domain == STOP_SIGNAL
```



```

        myStop()
    end
    D := domain
    F := f_init
    while not Steady_State
        [SEL]
    end
end

```

where, myStop() is a user-defined function which terminates the kernel. But, before, if needed, it: (i) saves results, (ii) propagates the stop signal; (iii) notifies the connected conduits.

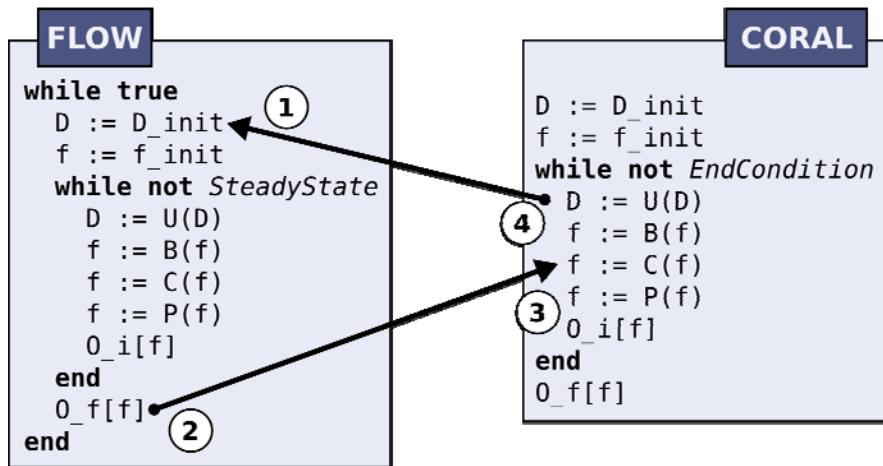


Fig. 4. Coupling strategy for the coral growth CxA.

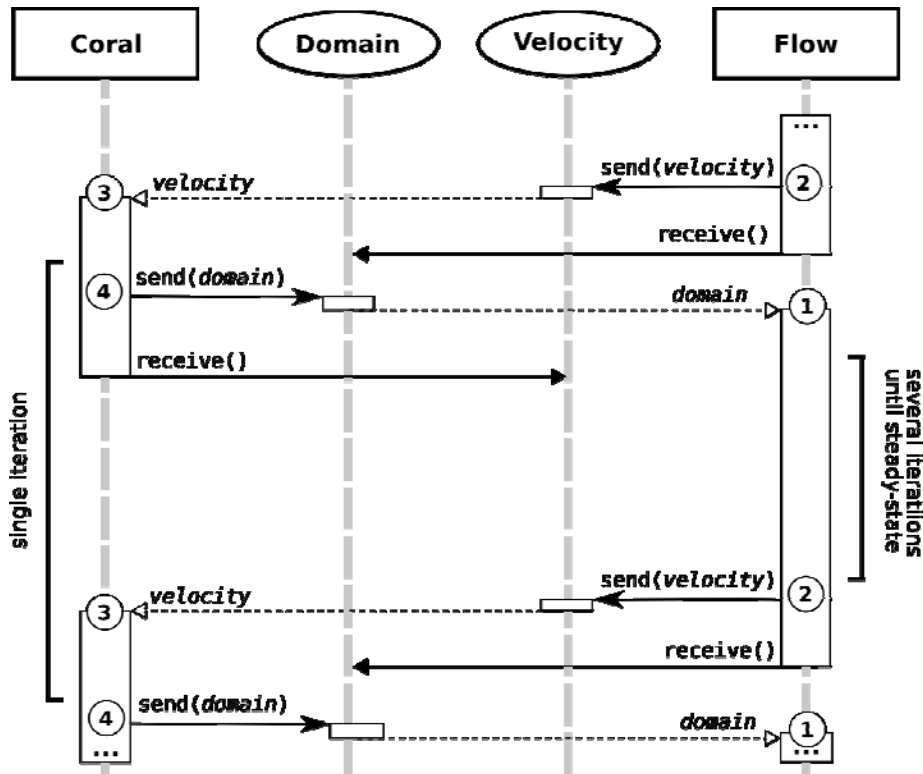


Fig 5. UML sequence diagram of the CxA shown in Fig. 4. The vertical lines represent the “life line” of the process: the kernels are represented by rectangles and the conduits by ovals. When a process is active, the gray life line is replaced by a vertical white rectangle. The arrows represent interaction. Solid arrows with triangular heads are blocking interactions and solid arrows with stick heads represent non-blocking interactions. The return values are indicated by dashed arrows. The circled numbers correspond to Fig 4.

1.6 Scale-splitting error

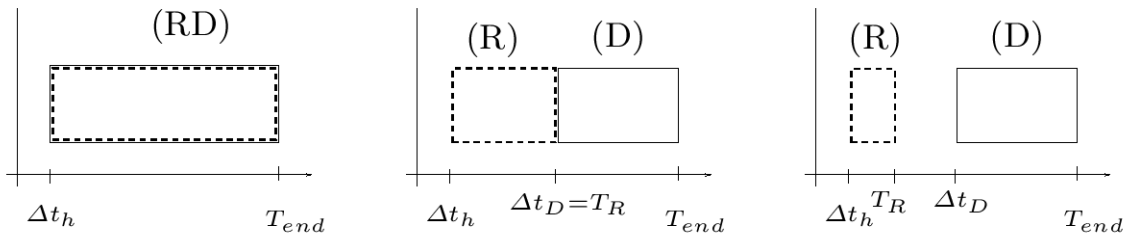
In this section we present a mathematical study of the impact of splitting a MM problem into two single-scale sub models. By reducing the range of scales for the full problem, we expect a significant speedup in terms of execution time. But we also expect a loss of accuracy. The CxA formalism can be used to express, through the SEL operators, the difference between the full, multiscale computation and its approximation as the coupling of two single-scale models.

For the case of a reaction-diffusion (RD) process for a concentration field $\rho(t, x)$ obeying

$$\partial_t \rho = d \partial_{xx} \rho + \kappa(\rho_\lambda - \rho), \quad t \in (0, T_{end}), \quad x \in (0, L)$$

explicit calculations can be done. The problem is multiscale because we assume that the reaction is much faster than the diffusion. So, intuitively, updating the diffusion at the same time resolution as the reaction is a waste of CPU resources. On the other hand, enforcing a different time scales for the two processes will involve an error.

The SSM for the RD process is given below in three cases. Left: the full problem; middle: the case where the reaction is run for a time equal to the diffusion time step; right: for a case (not considered further here) where the reaction would reach a steady state in a time faster than the diffusion step.



Here we are interested in splitting the problem as shown in the middle diagram. We only sketch the main steps of the mathematical analysis. We refer the reader to [Caiazzo 2008] for a full account of the derivation.

In terms of the state variables f , the full-scale problem can be expressed in a LB approach as

$$\hat{f}_h^{t+\Delta t} = P_h(I_h + \Omega_{Dh} + \Omega_{Rh})\hat{f}_h^t = \Phi_h \hat{f}_h^t,$$

where h indicates the finer scale of the problem (here we set the lattice spacing to scale as h and the time step as $h \cdot h$). I is the identity operator and P the propagation operator defined as

$$(P_h \hat{f}_h)_i(j) = \hat{f}_{i,h}(j - c_i),$$

where i labels the possible lattice velocity c_i . The reaction and diffusion operators Ω_R and Ω_D are defined as

$$(\Omega_{Rh} \hat{f}_h)_i = \frac{h^2}{2} R(\rho(\hat{f}_h)), \quad (\Omega_{Dh} \hat{f}_h)_i = \frac{1}{\tau} (f_i^{eq}(\hat{f}_h) - f_{i,h})$$

for a prescribed function f^{eq} .

Simple algebra shows that $f^{eq}(\Omega_R(\hat{f})) = \Omega_R(\hat{f})$. Therefore

$$\forall \hat{f}_h \in F_h, \quad \Omega_{Dh} \Omega_{Rh} \hat{f}_h = 0. \quad \text{and} \quad \forall \hat{f}_h \in F_h, \quad \Omega_{Rh} (1 + \Omega_{Dh}) \hat{f}_h = \Omega_{Rh} \hat{f}_h.$$

We can then split the LB algorithm in the equivalent form

$$\hat{f}_h^{t+\Delta t} = P_h(I_h + \Omega_{Dh})(I_h + \Omega_{Rh})\hat{f}_h^t = D_h R_h \hat{f}_h^t$$

separating reaction $R_h = I_h + \Omega_{Rh}$ and diffusion $D_h = P_h(I_h + \Omega_{Dh})$.

We now define a Complex Automaton, composed of two Automata CA_R (reaction) and CA_D (diffusion), introducing a coarser time step for the diffusion model, i.e.

$$\Delta x_D = \Delta x_R = h, \Delta t_D = M \Delta t_R = M h^2,$$

for a natural number M . In practice, it corresponds to executing M steps of the reaction (CA_R), followed by a single diffusion step. In this case the CxA model is analogous to an operator splitting approach. To describe the CxA model formally, we introduce a vector of parameters $H = (h_R, h_D)$, with the corresponding discretizations. We describe the evolution of the system with state variable $\hat{f} = (\hat{f}_R, \hat{f}_D)$. As the two processes act in the same domain (we have a sD problem), it is possible to write the algorithm only depending on \hat{f}_D as

$$\hat{f}_{D,h}^{t_n+\Delta t} = D_h(\tau_D) R_h^M \hat{f}_{D,h}^{t_n}.$$

Where τ_D determines the diffusion coefficient d and scales with M as $\tau_D = \frac{1}{2} + Md$. The scale splitting error (i.e. the difference between the full scale model and the two coupled submodels) reads

$$\begin{aligned} E^{A-CxA}(M) &:= \left\| \rho \left((D_h R_h)^M - D_h(\tau_D) R_h^M \right) \right\| \\ &\leq \left\| \rho \left((D_h R_h)^M - D_h^M R_h^M \right) \right\| + \left\| \rho \left((D_h^M R_h) - D_h(\tau_D) R_h^M \right) \right\| = \\ &= E_1(M) + E_2(M). \end{aligned}$$

The contribution E_1 can be estimated by the commutator $[D_h, R_h] = D_h R_h - R_h D_h$. It can be shown that $[D_h, R_h] \hat{f}_h = O(h^3 \kappa \partial_x (\rho(a_h) - \rho_\lambda))$.

By counting the number of times the commutator appears in the difference $(D_h R_h)^M - D_h^M R_h^M$, we have (by induction)

$$(D_h R_h)^M - D_h^M R_h^M = O(M^2 h^3 \kappa)$$

The term E_2 derives from the time-coarsening of the diffusion part of the original lattice Boltzmann algorithm. Analyzing a diffusion depending on the modified relaxation time we obtain $\rho(D_h^M - D_h(\tau_D)) \hat{f}_{D,h} = O(M^2 d^3 h^2)$.

In conclusion, the splitting error for the reaction-diffusion process can be expressed as a function of the enforced time separation parameter M as

$$E(M) \leq O(M^2\kappa) + O(M^2d^3).$$

The above derivation can be validated on the exactly solvable problem

$$\partial_t \rho = d \partial_{xx} \rho + \kappa(\rho_\lambda - \rho), \quad t \in (0, T_{end}), \quad x \in (0, L)$$

with periodic boundary conditions in x , with $\rho_\lambda(x) = \sin(\lambda x)$, for $\lambda \in 2\pi\mathbb{Z}$. The analytical solution is

$$\rho_\lambda^*(t, x) = \exp((-4d\pi^2 + \kappa)t) \sin(2\pi x) + \frac{\kappa}{d\lambda^2 - \kappa} \sin(\lambda x)$$

The physical scale separation is specified by the non-dimensional

$$\text{parameter } \sigma = \frac{\kappa}{\lambda^2 d}.$$

In Fig. 6 we show the results for the scale-splitting error as a function of M , fixing h and κ , for different values of d . This choice is due to the fact that the grid size is related to the reaction rate for stability reasons. In general, for better scale separation (larger values of σ) we obtain lower scale-splitting error.

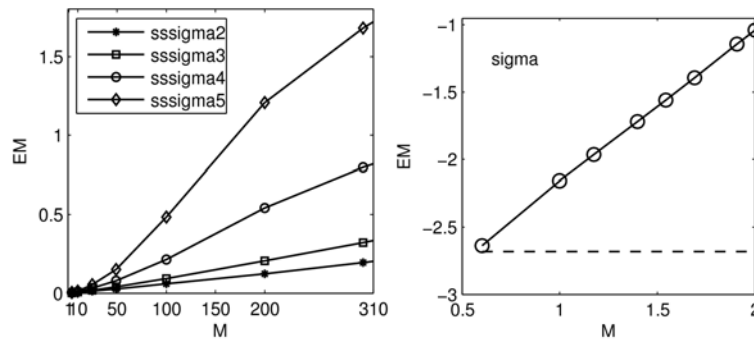


Fig. 6: **(left)**: Scale-splitting error $E(M)$ as a function of M for different values of σ ($h=0.02$, $\lambda=4\pi$, $\kappa=10$, $d=\{0.5,1,2.5,5\}$). **(right)**: Error in log scale for a range of moderate M , compared to the error of the original (fine-discretized) algorithm (dashed line) with respect to the exact solution).

Fig. 6 (right) compares the scale-splitting error as a function of M (for moderate values) with the error of the full scale LBM with respect to the exact solution. We note that for small M , the scale-splitting error is of the same order as the discretization error, i.e. the loss of accuracy of the CxA model is comparable with the numerical error produced by the original algorithm.

1.7 Summary of the CxA methodology

Tasks 2.1 to 2.4 resulted in the foundation of our CxA framework. So far, we have identified the main components of a CxA theory. A CxA is based on (1) the Scale Separation Map (SSM), (2) a generic Submodel Execution Loop (SEL), (3) coupling templates between the operators of the SEL, (4) smart conduits which implement these couplings transparently and, (5) the asynchronous, distributed data-driven execution model.

Within the CxA formalism, different strategies can be defined to split a full multiscale problem into several coupled single-scale submodels. These strategies can be expressed in terms of the SEL operators, thus giving a way to analyze mathematically the validity of a given decomposition (or scale partitioning).

Therefore, a coupling translates into a (sometimes intricate) product of operators. When applied to a specific problem, this operator expression can be compared with that of the full model, which has not been split into single-scale submodels. As a result, one may expect to quantify the accuracy of the CxA to represent a given multi-scale system. This route has been successfully applied to a reaction-diffusion Lattice Boltzmann model in which the time-splitting methodology has been used to separate the scales. A scale-separation measure can be defined, from which the gain in execution time and the loss of accuracy can be parametrized.

The above line of research will be continued in the last year of the project. We will describe more representative applications with the operator formalism and describe mathematically the CPU gain and accuracy loss of a given scale partitioning. An important question will be to what extent the formalism allows generalization in terms of generic Coupling Templates.

In another future step, a CxA modeling language will be proposed, in the same spirit as the UML approach gives a description of a classical computer program. The first object of the CxA language is a rectangle located on the SSM. Each rectangle represents a sub-model. The size and location of a rectangle specifies the range of spatial and temporal scales that are resolved by the corresponding sub-model. Arrows can then be added to represent the coupling between sub-models and the smart conduits. The extremities of these arrows are labeled with the

operators of the SEL that are involved in the corresponding coupling. Furthermore, extra information can be added on top of each arrow to specify the data crossing the conduit.

This graphical language is an efficient way to describe a multiscale, multi-sciences application. It allows researchers to discuss multiscale models and specify the type of coupling with no ambiguities, even though they come from different fields of expertise.

Sction 2: The In-stent restenosis as a CxA

A fundamental step towards the validation of the Complex Automata simulation technique consists of the practical path to be followed, starting from the Multiscale Process, to construct the Scale Separation Map (SSM), and to build a Complex Automata (CxA) simulation.

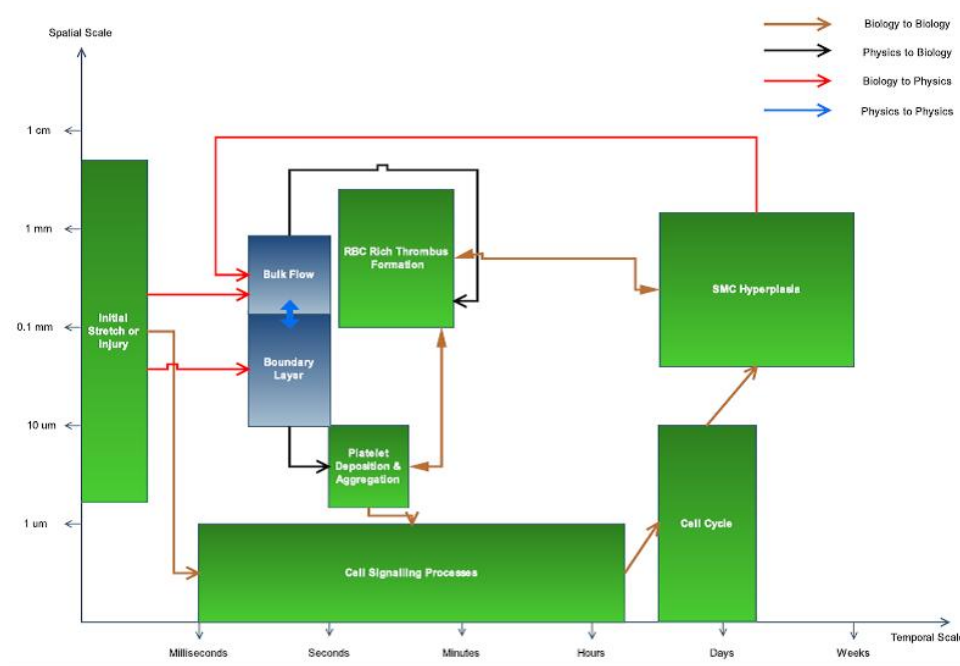


Figure 7: Simplified version of SSM for ISR, from Deliverable D4.1

Figure 7 shows the SSM for the COAST prototype application, the so called in-stent restenosis (ISR), where physical and biological processes have been placed according to the characteristic temporal and spatial scales. This SSM was presented in deliverable D4.1, and was based on an extensive literature survey as presented in Appendix B of D3.1. Please note that this version does not contain drug elution, but shows the starting point of the development of the CxA model.

Below intramural diffusion will also be accounted for, being a major process to model the drug elution.

The identification of the relevant processes and their placement of the SSM turned out to be far from trivial, and the SSM as such is still under debate and being updated. This is due to ongoing research in restenosis, new insights, etc. However, such SSM is just the starting point of a CxA model. In the following section we discuss how a CxA model for ISR can be set up in practice. We first formulate a generic methodology to specify a CxA model, based on a SSM, and then apply this to the case of ISR. We also touch upon issues related to implementation of the CxA model as a CxA simulation. However, actual implementation of the model is part of WP3 (task 3.5) and results obtained so far will be reported in D3.2.

2.1 Towards the computational model

In general, the following steps are needed to go from a multiscale process to the full specification of a computational CxA model:

1. Identify the computational domain of the original problem and the initial conditions.
2. Identify and enumerate the relevant single scale sub-processes and their scales (as in figure 7).
3. Specify the computational models for the sub-processes. Identify their computational domains and initial conditions. Note that at this stage, some sub-processes can be grouped in a single model, so that the actual SSM, used for the CxA model can be slightly different from the SSM in figure 7.
4. Draw a Scale-Separation Map (SSM) where the selected algorithms for the single scale models are placed according to their discretization parameters.
5. Identify the interaction between processes. Specify whether we are dealing with single Domain (sD) or multi Domain (mD) coupling, more in general, identifying the coupling templates (to be later mapped to specific smart conduits).
6. Specify the Connection Scheme.

2.2 Multiscale Problem

2.2.1 Definition of the global Domain

First of all, we have to define the computational domain of the simulation. The investigation presented in [Duraishwamy et al., 2007] concludes that a full 3D model of the stented coronary artery is

necessary, in order to include curvature effects which are relevant in the development of the restenosis (see also Figure 8).

However, currently we do not have full 3D information available, especially with respect to the curvature of the stented vessels (see also discussion in deliverable D4.2 on the analysis of the available histological sections) for geometry and validation. Whilst stents are designed to have some axial flexibility they are often stiffer than the native vessel and thus have a straightening effect. One valid option is to model the stented area as a straight tube. In fact, this will be enough to validate the COAST ideas, the framework and the coupling templates.

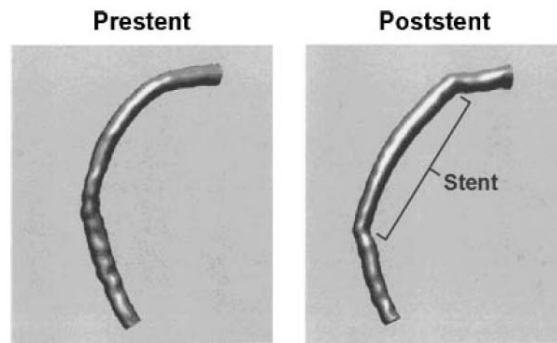


Figure 8. Three-dimensional reconstruction of a right coronary artery before and after stent implantation (taken from [Duraismamy 2007]).

Therefore, we will restrict to a cylindrical vessel of length 1.5 cm, with a diameter of 2 mm. Stent struts have a size of the order of $100\mu\text{m}$. The vessel wall tissue has a thickness of 0.3mm. The structure of the vessel wall, with internal elastic lamina, will be explicitly modeled as a physical barrier that, unless breached prevents smooth muscle cells from migrating into the lumen.

For simplicity, in the following section, we describe a CxA setup focussing on a 2D version, in order to have a clearer understanding of the difficulties which arise when building the CxA model. The two dimensional equivalent of the above 3D domain is a channel of dimensions 1.5cm x 2.6mm, where layers of tissue are present at the top and at the bottom. Stent struts are modeled as squares of $100\mu\text{m}$ (see Fig. 9).

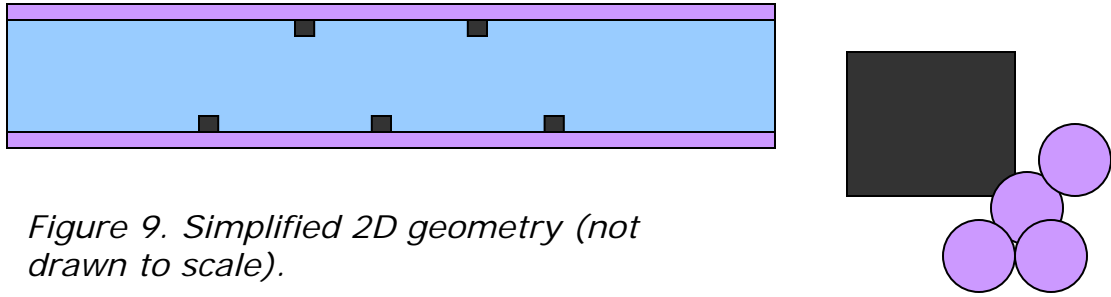


Figure 9. Simplified 2D geometry (not drawn to scale).

Smooth muscle cells have an average diameter of $30\ \mu\text{m}$. The typical size of red blood cells is $8\ \mu\text{m}$, whilst platelets are 4 or 5 times smaller.

2.2.2 Global Initial Conditions (IC)

In addition to the domain, we have to specify the initial condition (IC). IC consists of two parts: definition of the initial geometry (mainly needed for the flow solver used to simulate the blood flow) and definition of the initial injury. These two factors are strictly related, since stent geometry, mismatch, and deployment determine the characteristics of both the initial geometry and the injury. In the model for in-stent restenosis, as the initial condition the stent deployment is modeled, starting from a given stent geometry, and simulating a compression of the cellular tissue. This provides initial stresses in the cells and a model for the rupture of internal elastic lamina, with consequent level of injury score.

Initial geometry

The initial geometry is generated starting from a given stent geometry at rest (assumed to be known) and simulating a deployment into the tissue, see figure 10. More details are discussed further in deliverable D3.2 in the context of task 3.4 (single scale models) and 3.5 (implementation of prototype).

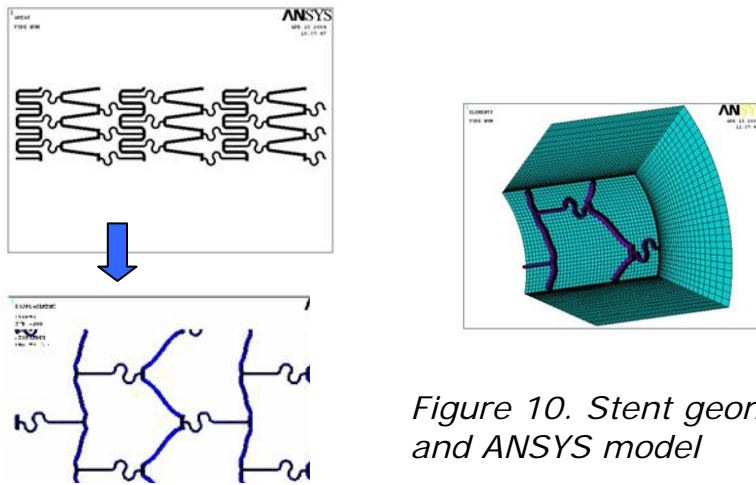


Figure 10. Stent geometry and ANSYS model

Initial injury

The state after the initial injury is defined as the state of the stresses after the stent deployment plus the injury score (see Fig. 11), i.e. of the location on the initial geometry and of the absence of endothelial cells (EC).

As yet, models including the injury score are not available. To begin with, we consider that endothelial cells (EC) are completely removed after stent deployment. D4.2 reports the presence of isolated patches of endothelial cells immediately post stenting but there is no information as to whether these cells are viable and able to function normally. Note however that the implementation of the IC (see task 3.5) will be such that we can easily accommodate for other IC, where patches of EC are still present. Injury score determines indirectly regions of the domain where the smooth muscle cells (SMC) are more prone to proliferate. This will be mainly determined by the rupture of internal elastic lamina, and specification of this is part of the initial geometry after stent deployment.

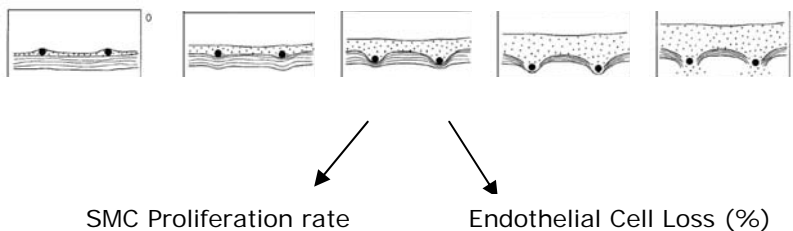


Figure 11. Diagram of initial injury (from WP3.2)

2.2.3 Single scale sub -processes

Now, we enumerate the relevant processes that we intend to simulate with dedicated single scale models:

1. Bulk Flow
2. Boundary Interaction/Transport Layer
3. Platelet Deposition, RBC-rich Thrombus Formation
4. Neointimal (SMC) Hyperplasia (Cell Cycle, Cell Signaling, Growth)
5. Drug Diffusion

Remark: The ECs are believed to play a quite important role in ISR (as discussed in detail in deliverable D3.1), but they are not explicitly modeled in the initial setup. We will explicitly include ECs in the neointima sub model in the next version of that sub model, scheduled to be delivered in Y3 of the project.

2.3 Computational description of single scale models

In this section, we describe each single scale model in terms of computational domain, resolution, sub-model execution loop, and coupling templates. For details of the single scale models and their adaptation for the COAST framework, we refer to deliverable D3.1 and D3.2, where the results of task 3.4 on the adaptation of single scale models are reported.

2.3.1 Bulk Flow (BF)

The bulk flow is simulated inside the lumen. The flow is periodic (pulsatile), driven by velocity inlet and pressure outlet conditions. The flow profile will be taken from the literature and adapted to the specific conditions prevailing in the generic pig model from which the histological sections analysed in WP4 (see deliverable D4.2) were obtained. No-slip boundary conditions are used on the vessel boundary and on the stent struts. We do not consider non-Newtonian blood rheology in the present model, assuming that this only gives small contributions to the final effect. However, as the flow solver is capable of simulating non-Newtonian blood rheology (as reported in deliverable D3.1), we intend to further test this hypothesis.

For the numerical simulation, we use a lattice Boltzmann Method (LBM). In order to resolve the flow patterns near the stent struts, the discretization must be sufficiently fine (see Fig. 12). This results in a relatively high computational effort, due to the fact that the struts are much smaller than the diameter of the lumen. The practical solution consists of employing refined meshes near the stent. However, for the moment, and in the present document, we have used a uniform grid. As discussed in deliverable D3.2, we do intend to address this problem by allowing for refined meshes in the flow computations.

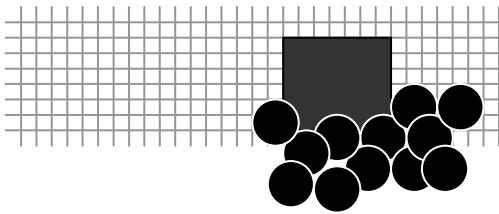


Figure 12: The computational domain with a strut and several smooth muscle cells

As input, this single scale model needs the mesh, generated from the initial geometry configuration, according to a given grid size.

2.3.2 Boundary Interaction/Transport Layer

Physical problem:

The Boundary Interaction/Transport Layer (TL) aims to model advection of RBCs and platelets driven by the blood flow, together with the deposition of these cells in specific regions of the lumen boundary (depending on the strut position).

We remark that the deposition influences another process, the formation of thrombus, which in the present document is considered as a separate box on the SSM. By separating transport from aggregation (and consequent thrombus formation), we separate the physical model from the biological one.

There are different modeling choices for the transport, which will be shortly discussed below.

Passive Transport

Transport of different species of cells can be modeled with a continuous approach, i.e. representing the cells with scalar concentration fields $c_m(x, t)$, obeying an Advection-Diffusion(-Reaction) (ADR) equation, where species are transported according to the flow velocity field. This solution is relatively flexible and simple from the implementation point of view. Additionally, in this particular situation we can use a lattice Boltzmann model for scalar transport, which can

be attached to the bulk flow solver. This corresponds to the use of the same spatial domain and grid discretization as the bulk flow.

Concerning the time scale, there are at this point at least three possible approaches, illustrated in Fig. 13:

- **TL.1:** Use the same time discretization for Bulk Flow and Transport, i.e. using the velocity field $u(t,x)$ computed by the flow solver at each time step.
Pro: a tightly coupling relation Transport→Fluid (for example, like concentration-viscosity) can be in principle included in the model.
Con: Need of flow solver continuously, i.e. relatively high memory consumption and computational overhead.
- **TL.2:** Assume that the feedback of Transport model on Bulk Flow is slow. In practice, this corresponds to using the same flow field for many flow cycles, as input for the Transport model. This idea relies on the assumption that the (variations in) concentration and any deposition, slowly change the shape of the boundary (note: this happens through thrombus deposition).
Pro: More efficient implementation, since the bulk flow solver is called fewer times than the ADR solver.
Con: Assumes scale separation in the sense that the viscosity of the fluid can be modeled independently. It might not be true in general. The model might be still expensive in terms of memory.
- **TL.3:** Use different time discretization for bulk flow and transport: complete time separation. Specifically, after having resolving the fluid over a cycle, we map the velocity on a coarser time discretization and update the concentration fields with the bigger time steps (multiple of a cycle). In the simplest case, the average of the velocity could be used for the update. A more rigorous argument is required to justify this methodology (See also [*Parmigiani 2008*]).
- **Pro:** Fastest options.
Con: Use of the average velocity field might be too simple. Including higher order moment might be a solution; however, depending on the configuration this might not reduce the computational effort if complex relationships need to be computed.

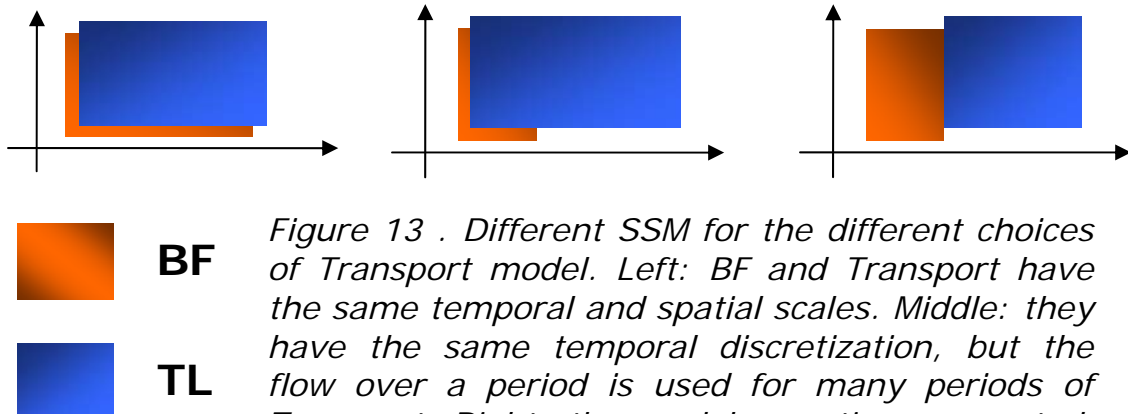


Figure 13 . Different SSM for the different choices of Transport model. Left: BF and Transport have the same temporal and spatial scales. Middle: they have the same temporal discretization, but the flow over a period is used for many periods of Transport. Right: the models are time separated. In all cases, the spatial grid is the same.

A more complex model could be considered, where the spatial dimensions are also considered in more detail, and where blood cells are fully resolved suspensions interacting with the flow. Below, we sketch the dimensions of RBCs and platelets, compared with a stent strut. A detailed suspension simulation yields a high computational effort and additional computational difficulties. However, we have pursued investigations of suspension simulations, as they may be used to compute probabilities of EC progenitors arriving from the blood stream at the vessel wall.

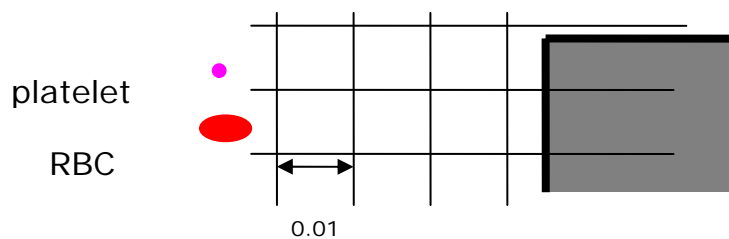


Table: Transport Layer model: size studies

	2D model	3D model	details
Fine Voxel volume	10^{-4}	10^{-6}	$\Delta x = 0.01$
# particles (RBC) per fine Vx (concentration ~ 50%)	~1.2	~1.8	radius= 0.008
Fine grid volume	6	6	0.3 x 20

# fine grid nodes	6×10^5	6×10^7	0.3 x 20
# particles (RBC) (region 0.3 x 0.3 near the strut)	$\sim 10^3$	$\sim 10^5$	0.3 x 0.3

Assuming that we have a suitable model for blood rheology, we could discard option **TL.1** and select **TL.2**, or the scale separated version **TL.3**. Possibly, a more precise or complex treatment of the phenomena can be implemented, also driven by the results obtained with this first version. We will further investigate this issue in Y3 of the project.

2.3.3 Platelet Deposition & Aggregation – RBC Rich Thrombus Formation (RTF)

This process aims to describe the thrombus formation, which characterizes the stage immediately following the stent deployment. The removal of endothelial cells and injury of the internal elastic lamina (IEL), triggers platelet activation (through release of chemicals or ‘activating agents’). Activated platelets are deposited on the region of injuring. Due to the flow field, and depending on the advection of blood cells near the injured region, a thrombus may appear downstream behind the struts. The platelets present in the thrombus are a source of chemical stimuli and signaling molecules, which influence SMC division and migration.

The transport layer described above will be the source of platelets to be deposited. Platelets are activated in a number of ways, by the injury, i.e. by the exposure of tissue after the deployment of the stent, and from the shear stress conditions of the blood flow. Activation occurs only in small regions close to the stent struts, where the vessel is injured. However, for practical reasons the model can cover the whole boundary, since they, once activated, can stick and aggregate in regions where the wall shear stress is low.

Long term dynamics Initial Conditions

Since these processes take place in a relatively short time frame, i.e. minutes (see deliverable D3.1 for an account of the temporal development of all processes involved in ISR) compared with the scale we are interested in (SMC hyperplasia and remodeling occurs over a period of days or weeks), a first very simplified, but yet effective, approach could be to use an initial guess for platelet deposition and initial thrombus, based on by experience, experimental results and

previous numerical experiments, to construct a thrombus (i.e. a new geometry) according to the struts.

From a biological point of view, we assume that the platelets immediately stick to regions without endothelial cells (EC), just redefining a smooth boundary for the flow.

The thrombus appears when RBC and platelets (becoming stickier due to the low flow stress) can deposit on regions close to the vessel boundary, especially in regions of recirculation. In practice we can

- run a flow simulation in the stented vessel geometry
- identify the regions where initial thrombus will more likely appear
- define new initial conditions for the long-term dynamics

With respect to the dynamics of the flow, these assumptions appear reasonable (in terms of an initial simplified model, which can subsequently be improved further), since small variations in the shape of the new geometry have a small influence on flow.

Biology-Biology Coupling and Thrombus Dynamics

The choice of long-term initial condition could be the simplest approach, but it brings with it a number of other issues. In fact, the thrombus is not only a geometric constraint. We have to include its influence on the other processes in the rest of the model.

For example, the thrombus influences the SMC proliferation, since platelets are sources of chemical signals. Assuming a signaling dependent on the platelet concentration in the initial thrombus, we have to

- estimate the thrombus composition
- define biological rules for the coupling platelets signaling → SMC growth;

To take into account what happens, mechanically, when the SMC tissue grows, we need to

- define an interaction between the growing tissue and the already present thrombus;
- update dynamically the concentration of platelets and the signaling rules

The issue of the variation in the thrombus shape is not only related to coupling with the SMC. In fact, platelets and RBC are constantly brought to, and removed from, the surface of the thrombus. It is a typical case of **Advection-Diffusion-Erosion** model (with **moving boundaries**).

As remarked before, a fully resolved simulation is probably not feasible, because *of the computational effort* and limits of *accuracy achievable*.

Snow-Transport Model with Immersed Boundary

Deposition, aggregation and erosion can be modeled using a “snow-transport” model [Massetot 1998, Chopard 2000, Dupuis 2002, Ouared 2005, Chopard 2006], which combines the values of cell concentrations advected by the flow with additional biological rules to decide whether the concentrations solidify, or the existing solid boundary is removed modifying the shape of the fluid-solid interface.

An important issue concerning the multiscale coupling in time now appears. Assuming that the time scale of the thrombus formation is much slower than the flow cycle, computational time could be saved if the flow simulation did not need to be repeated many times. In practice, this can be done in different way. Below, we describe briefly the possible approaches.

- **Increasing sticking/aggregation probabilities:** In this case, we *enhance* the dynamics of the thrombus, in order to approximate the state of the system after many flow cycles, only using the results after a single cycle. This corresponds to assuming a quasi-linear dependency of growth on these probabilities (probably true for a small probability.)
- **Projective integration method for boundary growth:** Removing the quasi-linearity assumption, a projective integration of the interface position for the interface position can be used. In practice, we assume that the evolution of the boundary in time is described by proper ODE, with a typical time scale slower than a flow cycle. This class of approach has been extensively described in literature within the framework of *Equation-Free* approaches. There is the practical problem of how to represent the interface computationally. Simply using solid voxels requires some extra care (see [Stahl 2008]), and using a smooth interpolation could lead to expensive algorithms, which reduce the pay-off of the multiscale coupling substantially.
- **Thrombus Growth via Immersed Boundary:** Immersed Boundary is a practical solution often used to avoid explicit representation of the interface. In this particular case, the

thrombus is represented by a discrete set of points along the vessel boundary, which can move into the lumen of the vessel according to the concentration of platelets located in the neighborhood and to other (biological/statistical) rules. The movement does not affect the flow boundary conditions directly, but only through a force (*immersed boundary force*) felt by the flow field, which in fact is used to approximate the no-slip velocity on the current thrombus position. Erosion can also be easily included in the model, allowing thrombus tracers to move backwards.

The first version of the ISR CxA model will not include platelet deposition or thrombus formation. A first rudimentary model, assuming long-term dynamics initial conditions, will be added in the next version. If time permits will this be enhanced further to include some form of biology-biology coupling and thrombus dynamics.

2.3.4 Neointimal Hyperplasia (SMC)

This single scale model simulates the growth of smooth muscle cells into the lumen. Although it is possible to separate cell signaling, cell cycle, and cell mechanics, these processes are currently considered parts of a single model.

We use an Agent Based Model where each agent represents a cell, which can act, depending on its internal state, on the input coming from neighbours (signaling) and on external factors (concentration of chemical factors, local stress values).

Cell mechanics is modeled by computing the forces on each cell and updating cell positions accordingly. This is iterated until an equilibrium configuration is obtained, thus solving the dynamic equilibrium equation by a local scheme. Forces on cells include repulsive (modeling incompressibility) and attractive (modeling adhesion) cell-to-cell forces, frictional resistance force, a random walk force, and internal forces. Cell movement is constrained by domain boundaries, solid walls representing stent struts and inner elastic lamina elements. Due to the current spatial scale sizes, the choice of using single agents for each cell appears more consistent.

Biological processes are modeled by a set of agent rules, by which cell agents modify the variables representing their state. A cell cycle rule governs the progression of the cell around the cell cycle, thus

regulating growth and proliferation. Cells can be either in G0 (quiescent), G1(growing), or S/G2/M(growing/mitosis) states, and transitions between states can take place depending on internal as well as external variables. An apoptosis rule lets a cell cease to exist when exposed to large stress. Chemotaxis and random cell movement are modeled by a migration rule, which calculates a “motility force” term that is fed to the mechanical part of the model.

Signaling between cells is at the present stage modeled by exchanging messages between nearest neighbours.

The domain for the SMC model is the whole geometry, excluding the parts occupied by stent struts. Initially, SMCs are confined to the wall region outside the vessel. However, they can move and proliferate into the lumen.

2.3.5 Drug Diffusion (DD)

Drug eluting stents can be used to reduce the proliferation of SMCs. In biological tissue, diffusion is a complex process because of the heterogeneous nature of the tissue. To model this diffusion process, we use a generic anisotropic diffusion law:

$$\partial_t \rho(x, y, z; t) = \partial_\beta D_{\alpha\beta} \partial_\alpha \rho(x, y, z; t) \quad (5.1)$$

Where ρ is the concentration of the drug in the position x, y, z at time t . $D_{\alpha\beta}$ is the 3x3 diffusion tensor and the indices α and β indicate spatial components. Einstein summation convention applies for repeated indices.

When the diffusion tensor eigenvectors are orthogonal to the spatial axis, D is a diagonal matrix. In other situations, the eigenvectors display an angle θ with the coordinate axis. The diffusion tensor can then be expressed as:

$$D = R(\theta)D'R(-\theta)$$

where D' is a diagonal matrix. Since matrix symmetry is preserved by rotation, D is always symmetric. Using this relation we can first define D' with the tangential and the radial diffusion coefficients and then obtain D by choosing the suitable angle θ .

2.3.6 Intramural diffusion

In our demonstrator application, the drug diffuses from the stent struts to the SMCs.

The domain for the Drug Diffusion Model coincides with the space occupied by SMC. The struts act as sources of drugs, while the boundary is common between flow and cells are considered sinks (assuming that drugs eluted into the lumen are continuously flushed away by the blood flow). In practice, after discretizing the whole model geometry, we classify the mesh points in

- tissue,
- source,
- sink,

which are treated differently during the computation.

The diffusion tensor is chosen such that the diffusion along the artery axis or tangentially to a cross section is at least 10 times higher than the diffusion along the radial direction [Hwang 2001, Levin 2004] (see Fig. 14).

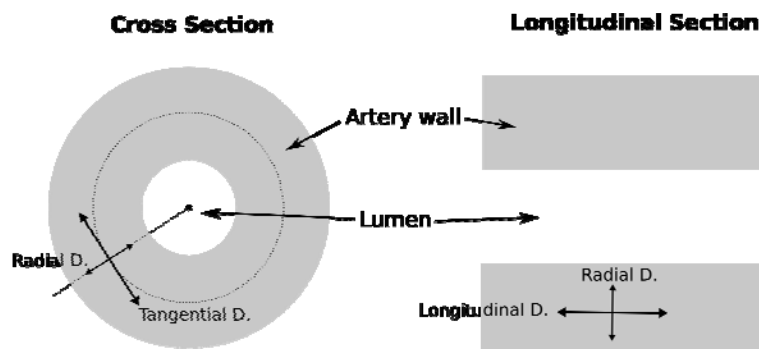


Fig. 14: Sketch of the 3D blood vessel geometry with the three diffusion directions: longitudinal, tangential and radial.

To solve equation (5.1) we had several possibilities. Instead of using a LBM approach to solve diffusion, we have chosen a Finite Differences (FD) approach solved by a Propagation-Collision loop. This choice was dictated by two arguments: (i) diffusion LBM use at least 4 times more memory than the FD; (ii) the choice of a FD helped us to investigate and demonstrate the coupling of a different approach.

Although we can solve the temporal evolution of the model, we are mainly interested in the steady state, because the scales are well separated. In [Levin 2004], the time scale to reach the steady state

ranges between 5 to 24 hours, depending on the drug (whether hydrophobic or hydrophilic in nature). On the other hand, the time step of the SMC growth is 1 day, thus validating our hypothesis of scale separation (also shown in Fig 15).

We can then compute the steady state by solving

$$\partial_{\beta} D_{\alpha\beta} \partial_{\alpha} \rho(x, y, z; t) = 0$$

2.4 Scale Separation Map

Taking into account the single scale models described and discussed so far, we show in Fig. 15 an updated Scale Separation Map for the current version of the Complex Automata Model

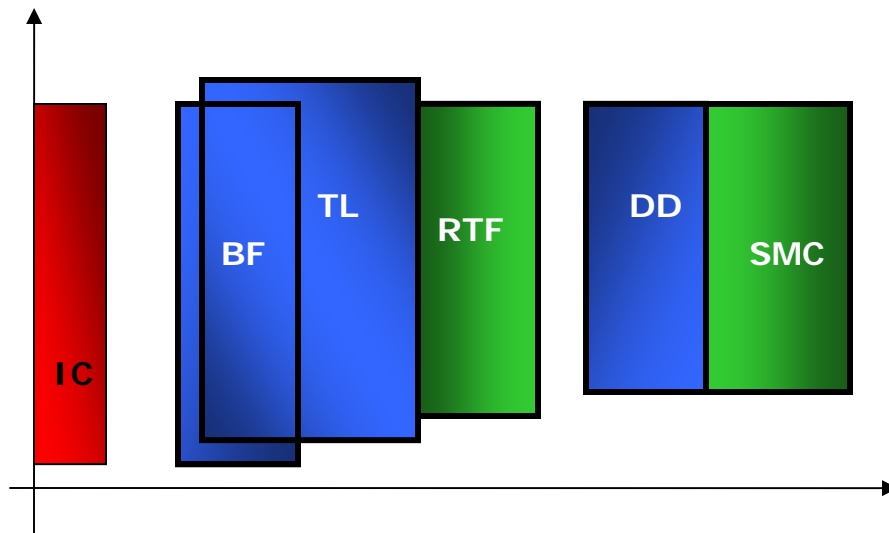


Figure 15: Updates SSM for the current version of the CxA model of ISR.

It contains 5 single scale models, plus an Initial condition box. Note that in this new version, we no longer have spatial separation. This is due to the fact that in the current version of the model, the SMC box explicitly models all the individual cells in the cell tissue, and the Bulk Flow is assumed to have a fine enough resolution to also discretize the Boundary layer.

Section 3: Conclusions

The CxA framework developed within the COAST project offers a new and general approach for multiscale, multiscale (MM) modeling. Provided that the problem under investigation can be scale-partitioned in several single-scale models, a significant execution speedup can be expected compared to the full scale simulation. Unfortunately some problems cannot be scale separated and for those, effective numerical simulations may be out of reach of current computers.

However, the proposed CxA framework is a well defined methodology to describe and specify a MM problem and naturally helps the user to propose a split in single-scale submodels. The generic execution loop, coupling templates, smart conduits and the execution model provide an efficient, transparent and reusable way to interconnect several subprocesses, whether new or from legacy work. The prototype application described in this document illustrates the methodology in detail.

Finally, the underlying CxA formalism can be translated into mathematical expressions and analyzed. In some situations, the error resulting from a scale splitting strategy can be explicitly described and controlled.

In the last year of the project we plan to consolidate the CxA theory in two main directions. The first is the continuation of the present work, with the goal of the mathematical formalization of more simple and representative applications, using the CxA constructs. The second is to use the CxA concepts, such as the scale map, coupling templates and SEL to define the element of a new multiscale modeling language. Such a language will be a natural way to specify a MM problem in an unambiguous way and will be of direct interest for task 2.6. In addition to theoretical development, the Integration task 2.5 will also consider the employment of COAST ideas and tools to other applications, such as, for instance, the transport and aggregation of magma fragments after a volcanic eruption.

Section 4: References

- [1] G. Agha. Actors: A Model of Concurrent Computation in Distributed Systems. MIT Press, (1986).
- [2] Bader D.A. (ed.): Petascale Computing: Algorithms and Applications. Chapman & Hall/CRC. (2008).
- [3] Alfonso Caiazzo, Jean-Luc Falcone, Bastien Chopard, Alfons Hoekstra. Error Investigations in Complex Automata Models for

- Reaction-Diffusion Systems. H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 260-267, (2008).
- [4] B. Chopard and M. Droz, Cellular automata modeling of physical systems, Cambridge University Press, (1998).
 - [5] B. Chopard, A. Masselot, and A. Dupuis. A lattice gas model for erosion and particles transport in a fluid. Computer Physics Communications, Vol. 129, 167-176, (2000).
 - [6] B. Chopard, Ouared, R., and Rüfenacht, D. A. A lattice Boltzmann simulation of clotting in stented aneurysms and comparison with velocity or shear rate reductions. Math. Comput. Simul. 72, 2-6 (Sep. 2006), 108-112. (2006).
 - [7] B. Chopard, J.-L. Falcone, R. Razakanirina, A. Hoekstra and A. Caiazzo. On the collision -propagation and gather-update formulation of a cellular automata rule. H. Umeo et al. (Ed) ACRI 2008, LNCS 5191,p. 144-151, (2008).
 - [8] A. Deutch and Sabine Dormann. Cellular Automata Modeling of Biological Pattern Formation. Birkhauser. Basel (2005).
 - [9] A. Dupuis and B. Chopard, Journal of Computational Physics 178, pp. 161-174, (2002).
 - [10] N. Duraiswamy, Richard T. Schoephoerster, Michael R. Moreno, and James E. Moore, Stented Artery Flow Patterns and Their Effects on the Artery Wall, Annu. Rev. Fluid Mech. 39:357–82, (2007).
 - [11] Hoekstra, A.G., Portegies Zwart, S., Bubak, M., Sloot, P.M.A.: Towards Distributed Petascale Computing. In: Bader, D.A. (ed.): Petascale Computing: Algorithms and Applications, Chapter 8. Chapman Hall/CRC. (2008)
 - [12] Chao-Wei Hwang, AB; David Wu, PhD; Elazer R. Edelman, Physiological Transport Forces Govern Drug Distribution for Stent-Based Delivery, Circulation 104:600 (2001).
 - [13] IEEE Comput. Sci Eng. 7, 14--53 (2005), Special issue on Multiphysics modeling.
 - [14] IJMCE: International Journal for Multiscale Computational Engineering, <http://www.edata-center.com/journals/61fd1b191cf7e96f.html>
 - [15] Andrew D. Levin, Neda Vukmirovic, Chao-Wei Hwang and Elazer R. Edelman. Specific binding to intracellular proteins determines arterial transport properties for rapamycin and paclitaxel. PNAS vol. 101, no. 25 9463-9467, (2004).
 - [16] A. Masselot and B. Chopard, Europhys. Lett. 42, 259-264, (1998).
 - [17] R. Ouared and B. Chopard, Lattice Boltzmann simulations of blood flow: non-Newtonian rheology and clotting processes, J. Stat. Phys, Vol 121 (1-2), pp. 209--221, (2005).

- [18] A. Parmigiani and B. Chopard, Investigating high Prandtl number fluid flows with a Passive Scalar Approach Thermal Lattice Boltzmann Model. EPJ, to appear, (2008).
- [19] SIAM Multiscale Modeling and Simulation, <http://epubs.siam.org/sam-bin/dbq/toclist/MMS>
- [20] Sloot, P.M.A., Hoekstra A.G.: Modeling Dynamic Systems with Cellular Automata. In: Fishwick, P. (ed.) Handbook of Dynamic System Modeling, Chapter 21. Chapman & Hall/CRC (2007).
- [21] P.M.A. Sloot; D. Frenkel; H.A. Van der Vorst; A. van Kampen; H.E. Bal; P. Klint; R.M.M. Mattheij; J. van Wijk; J. Schaye; H.-J. Langevelde; R.H. Bisseling; B. Smit; E. Valenteyn; H. Sips; J.B.T.M. Roerdink and K.G. Langedoen: Computational e-Science: Studying complex systems in silico. A National Coordinated Initiative. White Paper., February (2007b), <http://www.science.uva.nl/research/scs/papers/archive/Sloot2007a.pdf>
- [22] B. Stahl and B. Chopard, On the way to computer the wall shear stress in 1st order lattice Boltzmann boundary condition. Preprint Computers & Fluids, (2008)

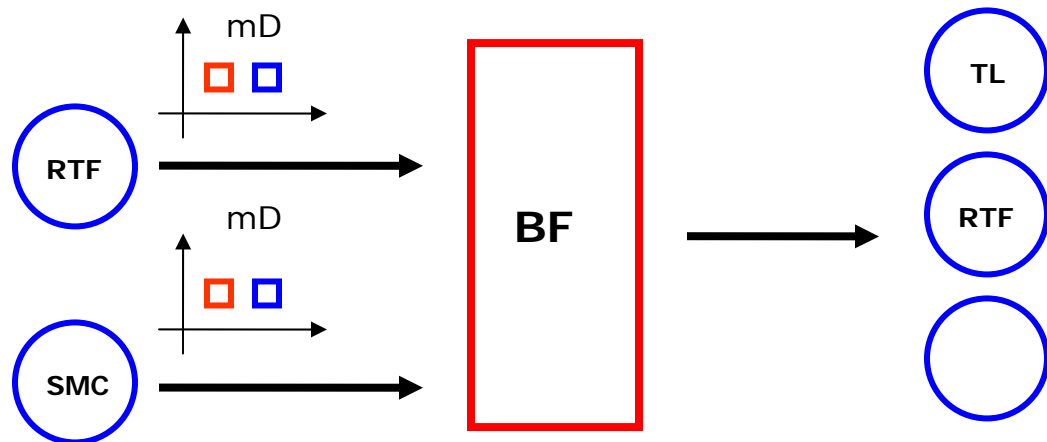
Section 5: Appendix Technical discussion for ISR modeling

This appendix holds a few more technical discussions on the Coupling Templates that will be applied in the model, discusses the coupling scheme that will be applied, and summarizes key parameters.

5.1 Coupling Templates

Details of the interactions between single scale models must be now specified, according to methodology and terminology adopted for the CxA theory. For each process, we describe the incoming arrows, specifying the type of coupling, the relative position of the coupled processes on the SSM, and the way the interaction acts in the sub-model execution loop (through boundary, collision, domain update, etc.). In the pictures below, the outgoing arrows are only used to indicate which observable is taken from the process. Above each coupling, a small iconic version of the mutual position on a SMM of both processes is drawn, in accordance with the Taxonomy, discussed in Fig. 2.

5.1.1 Bulk Flow



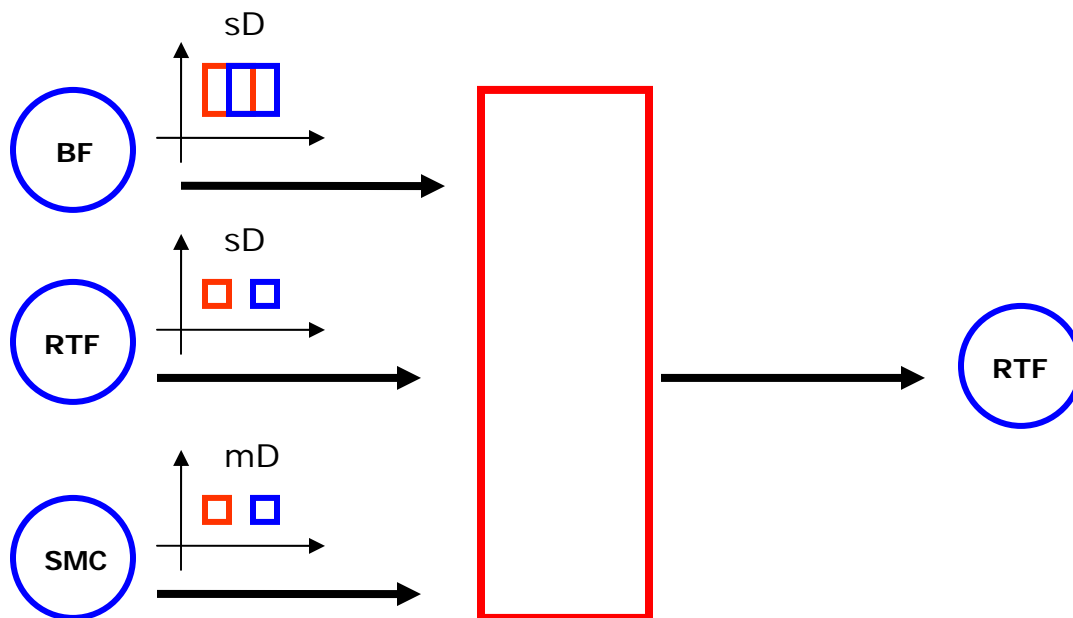
Incoming arrows. Bulk flow receives as input

- From SMC proliferation: current list of cells
- From Thrombus Formation: current thrombus domain.

In both cases, the coupling is multi-domain, and multiscale in time.

Outgoing arrows: as output, we need the fluid fields, used for the Transport Layer, the Thrombus formation, and the SMC Hyperplasia. In case of the transport layer, time-resolved fields are needed. For Thrombus formation and for SMC proliferation, average fields over a systole are needed. For the SMC, average wall shear stress is needed, whereas thrombus formation needs input on shear stresses in the whole domain where thrombus can occur.

5.1.2 Boundary Interaction/Transport Layer

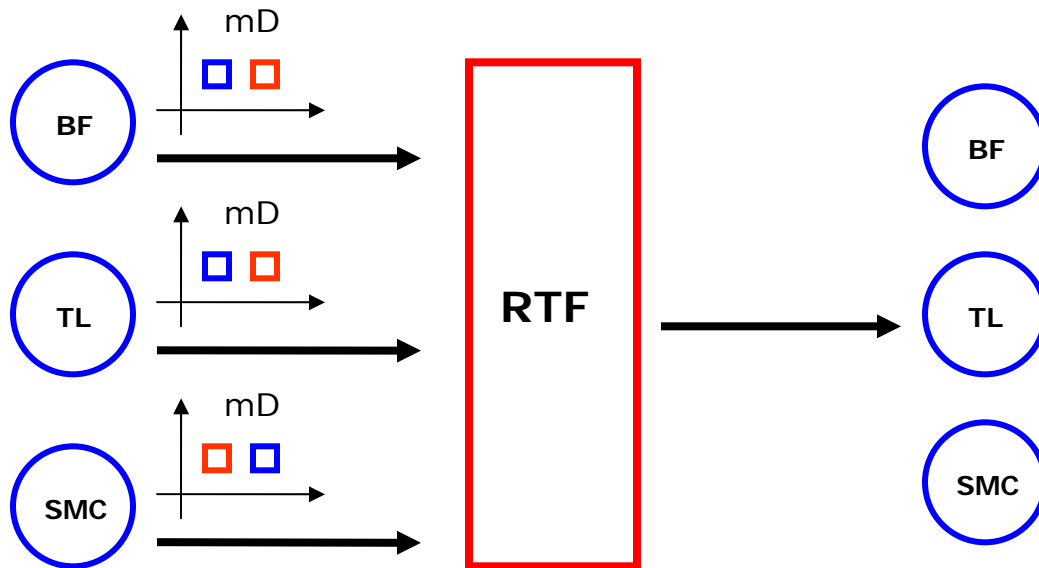


Incoming arrows. Boundary Interaction receives as input

- From Bulk Flow: time resolved velocity. It is a single domain coupling, through collision operator.
- From thrombus formation: current domain
- From SMC proliferation: current cell positions (domain)

Outgoing arrows. The concentration field is taken as output. It determines the aggregation and the thrombus formation.

5.1.3 RBC Rich Thrombus Formation



Incoming arrows.

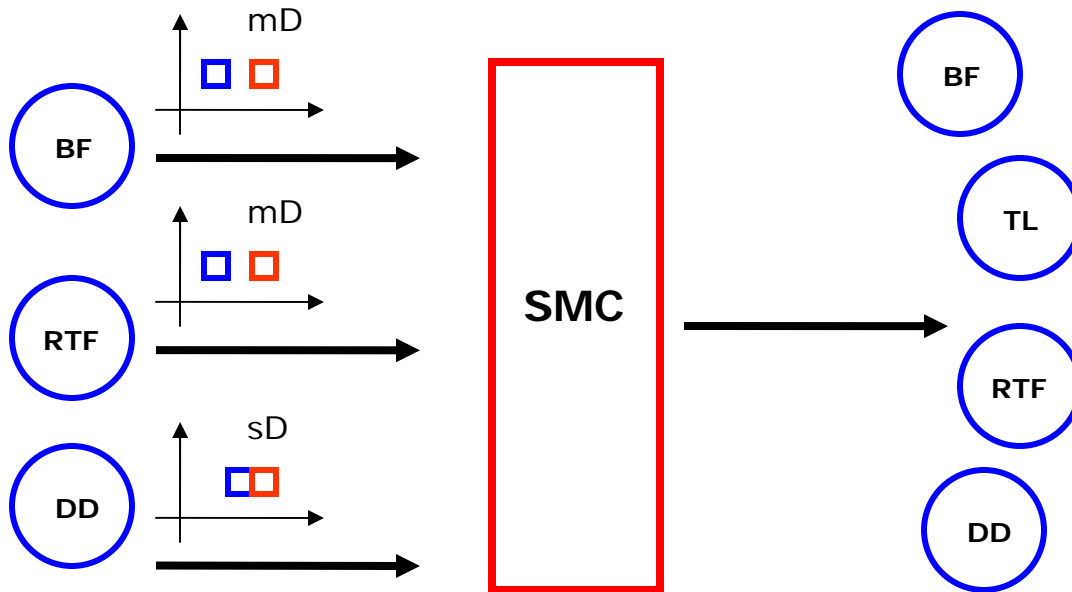
RBC-rich Thrombus Formation receives as input

- From Bulk Flow: velocity and stresses, averaged of a systole, which influence platelets activation
- From Transport Layer: RBC/platelet concentrations
- From SMC Proliferation: current cell positions (domain)

Outgoing arrows.

The thrombus modifies the domain (and the BC) of bulk flow and the boundary transport layer, and the thrombus formation influences the SMC proliferation.

5.1.4 Neointima Hyperplasia

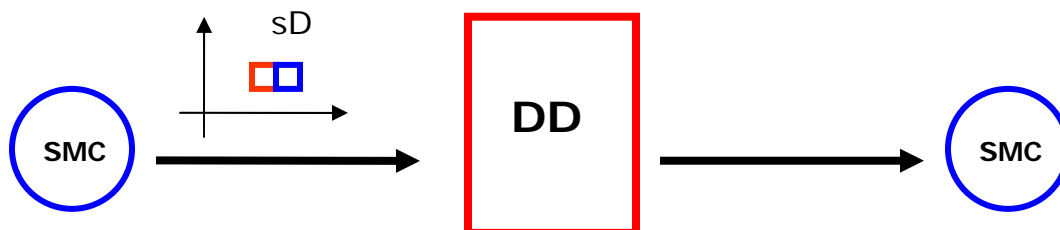


Incoming arrows. As input, the SMC needs:

- From Bulk Flow: averaged pressure and stresses on the boundary. It is a multidomain coupling, through collision operator, multiscale in time.
- From Thrombus Formation: concentration of platelets, state of the thrombus. It is a coupling through collision operator. Moreover, assuming that the thrombus evolves in the very first stage, this coupling can be probably included as an external factor, time-independent.
- From Drug Diffusion: drug concentrations per cell. Coupling multiscale in time, single domain, and through collision operator.

Outgoing arrows. SMC proliferation influences (through domain) the Bulk flow (1), the Transport layer, the Thrombus formation and Drug Diffusion.

5.1.5 Drug Diffusion



Incoming arrows.

- From SMC proliferation (4): current position of the SMC.

Outgoing arrows. Drug concentration for each cell.

5.1.6 Connection Scheme

Once the coupling templates have been specified, we can complete the information contained in the SSM with a Connection Scheme, which drives in practice the CxA simulation. From this point, the practical implementation of the CxA starts. See Deliverable 3.2 for details concerning the current CxA setup for the prototype application (figure 16)

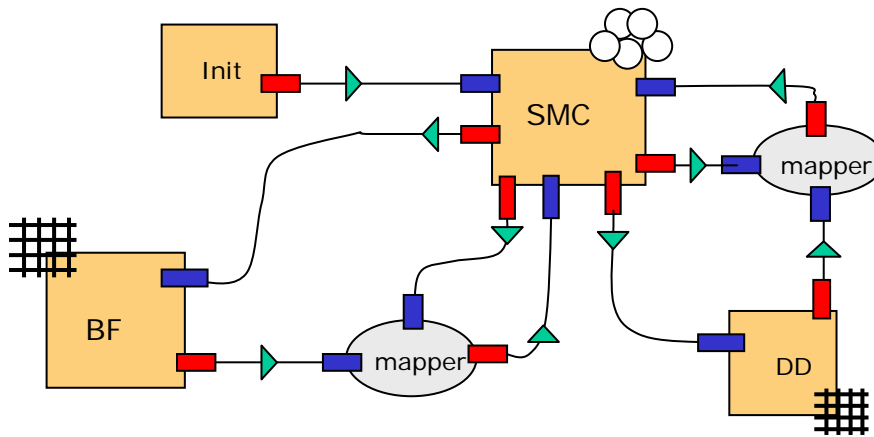


Figure 16: Example of Connection Scheme for a CxA coupling BF, SMC and DD (see technical deliverable 3.2). For each single scale model, it is indicated whether it is based on a lattice or on agent. Mappers are used to map different inputs onto the time dependent domain of cells.

5.2 Geometrical details of the model

length	15.000mm
diameter	2.000mm
strut	0.100mm
SMCs	0.030mm

length	15.000mm
RBCs	0.08mm
platelets	0.002mm

Relevant parameters of the model problem

Lumen diameter: 2-3mm
 Segment length: 1.5cm
 Tunica Inima thickness: 150 μm
 Tunica Media thickness: $\sim 160 \mu\text{m}$ (3-4 layer of SMC)
 Tunica Adventitia thickness:
 Platelet diameter: 1.5-3 μm
 Platelet density: 40%
 RBC Rich Thrombus: from 100 μm (around 50 platelets)
 SMC Diameter: 30 μm
 Strut diameter: 100 μm

Discretization parameters (indicative)

Bulk flow:
 (coarse grid) $\Delta x = 0.2\text{mm}$
 (refined grid) $\Delta x = 0.05 \text{ mm}$
 $\Delta t = 0.0001 \text{ s}$

Transport Layer:
 (coarse grid) $\Delta x = 0.2\text{mm}$
 (refined grid) $\Delta x = 0.05 \text{ mm}$
 $\Delta t = 0.0001 \text{ s}$