A Layered Behavioural Model of Platelets

Steve Schneider
Department of Computing
School of Electronics and Physical Sciences
University of Surrey
S.Schneider@surrey.ac.uk

Ana Cavalcanti
Department of Computer Science
University of York
Ana.Cavalcanti@cs.york.ac.uk

Helen Treharne
Department of Computing
School of Electronics and Physical Sciences
University of Surrey
H.Treharne@surrey.ac.uk

Jim Woodcock
Department of Computer Science
University of York
Jim.Woodcock@cs.york.ac.uk

Abstract

There is great interest in the application of nanotechnology to medicine, but concerns for safety are paramount. We present a modelling technique based on CSP and B as a starting point for simulation of networks of nanorobots. The model and the simulations are central features of our proposed approach to the construction of safety cases for nanomedicine applications, and complex networks of cooperating components in general. Our work is based on a case study: the clotting behaviour of (artificial) platelets. We present a model, and discuss its analysis and uses.

1. Introduction

With the renewed interest in nanotechnology [5], researchers have begun discussions about nano-scale robots: *nanites* [6]. In the safety-critical area of nanomedicine [7], it is proposed that nanites may be used as mechanical blood cells: artificial erythrocytes, phagocytes, and platelets. None of these exotic machines exists yet, but it is predicted by the Foresight Institute that we will see experimental nanites within the decade (`www.foresight.org`).

We present a first step towards a development technique that supports the construction of safety cases for networks of nanites. It is based on the abstract modelling of a design, which is validated through model checking, and subsequently refined to an executable program that is suitable for simulation. Our case study is a network of artificial platelets [8], which could staunch blood flow from a wound three orders of magnitude faster than natural ones.

The artificial platelets would carry a folded fibre mesh on board, to be used near an injured blood vessel. In such a situation, a protective film on the surface of the mesh would dissolve, revealing sticky sections that would bind a mat across the cut, immediately trapping blood cells, including other platelets, and stopping the bleeding.

Protocols are needed to guarantee that the artificial platelets release their meshes only when required to do so: if they do it at the wrong time, or in the wrong place, then the result will be an artificial thrombosis. The platelets would take their own decisions about when to deploy their meshes based on local information: there is no central control. Communication is subtle, and quite unlike that in computer networks: it is based on natural mechanisms. Platelets detect the presence of chemicals released by a wound; communication is only between neighbouring platelets.

We present here a simple CSP || B [15] model for platelets, motivated by the clotting process, called *haemostasis*. In CSP || B, B [1] machines are taken as communicating abstract data types, with CSP [12] controlling the way in which the operations are used. The two notations are kept syntactically and semantically isolated. As a result, it uses specific system architectures, and enables the direct application of the existing tools for CSP and B.

We build an abstract CSP || B model of an (artifi-

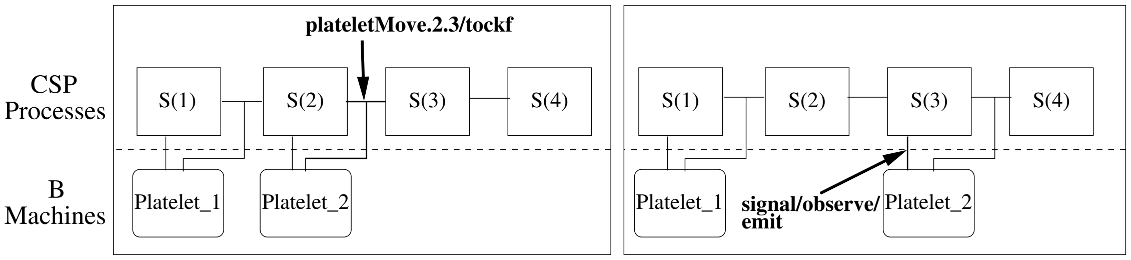

Figure 1. Architecture and movement of single platelet

cial) platelet. We propose protocols for bleeding detection, and initiation and termination of platelet clotting. We also capture location and mobility properties of platelets. A simple approach to time is used to model the blood flow, which is different in arteries and veins, for example. The model is surprisingly subtle and complex, with localised control and isolation of different aspects of the problem imposing interesting challenges. Our solution is a layered architecture.

In the validation of this model, use of the CSP model checker, FDR [9], and its associated animator, Probe [10], was essential. Our analysis of small networks of co-operating platelets and their protocols, in order to investigate their joint behaviour, is also briefly discussed here.

In the next section, we present a description of CSP || B, including the features of CSP and B used here. Section 3 presents our model: its detailed architecture and more interesting components. A discussion of our results comes in Section 4. Finally, we conclude in Section 5 with some discussion of related work, and our future directions.

2. CSP || B

CSP || B is an approach to combining the process algebra CSP with the formal development method B. The approach can be used to specify a complex system which is made up of two separate specifications: a number of CSP process descriptions and a collection of B machines (see Figure 1). Our aim when using CSP || B is to factor out "data-rich" aspects of a system into B machines so that the CSP descriptions focus on describing processes and their patterns of interactions.

A *machine* in B is a specification construct which encapsulates some variables and provides operations that query and manipulate those variables. In order to demonstrate that a machine is consistent, its initialisation must establish its invariant, and its operations must maintain it. The invariant is a predicate which constrains the variables to be of appropriate type and often defines constraints on the relationship between them. Operations take the form `x <-- c(v)`, where `c` is the operation name, `v` is a list of input variables, and `x` is a list of output variables. In Figure 5, we present a simplified model of a platelet defined in terms of a B machine. In our abstract model we consider several platelets and so the machine is replicated many times.

CSP is a language for describing processes of concurrent systems and their patterns of interactions. The unit of interaction is an *event* which processes perform and on which they may synchronise. Events can be unstructured (such as *enter*), or they can have some structure, generally of the form of a channel name c and some values v that are passed along that channel. Thus, the occurrence of $plateletMove.i.j$ can be understood as passing the values i and j along the channel *plateletMove* (provided i and j are of the appropriate type), and we use it to signify a movement of a platelet from one site to another. The *occurrence* of events is atomic.

Simple processes can be generated from the following syntax:

$$P ::= a \to P \mid c?x!v \to P(x) \mid b \,\&\, P \mid$$
$$P_1 \,\Box\, P_2 \mid P_1 \sqcap P_2 \mid S(i)$$

The process $a \to P$ can perform an event a and then behave as P. The process $c?x!v \to P(x)$ can accept any input x and output v along channel c, and having accepted x, it will behave as $P(x)$. The behaviour of the guarded process $b \,\&\, a \to P$ depends on the evaluation of the boolean b: if true, it behaves as P; if false, it is unable to perform any events.

The external choice, $P_1 \,\Box\, P_2$, is initially prepared to behave either as P_1 or as P_2, with the choice being made on occurrence of the first event. The internal choice $P_1 \sqcap P_2$, is able to behave as P_1 or as P_2, but this choice is made internally: the environment of the process has no influence over this choice. There is also an indexed form $\bigsqcap_{i \in I} P_i$ which provides an internal choice over a set of processes P_i indexed by I. $S(i)$ is a process name where i is an expression; this is used when we define the behaviour of an arbitrary platelet site.

In addition to the language for simple processes, CSP provides a number of parallel composition operators which can be used to combine processes. The operators we are concerned about in this paper are the following: $P_1 \ |[A\,|\,B]|\ P_2$ and $\|_{i \in I} P_i$.

The *parallel composition* operator, $P_1 \ |[A_1\,|\,A_2]|\ P_2$, executes P_1 and P_2 concurrently, requiring that they synchronise on events that are common to both their alphabets. The alphabet of process P_1 is denoted by A_1 and is the set of events that it can perform; similarly for P_2 and A_2. This composition operator is used when composing different platelet constraints together. The alphabet can be dropped when it's not implicit.

The indexed form of $\|_{i \in I} P_i$ allows us to construct combinations of similar processes. For example, when building up a collection of processes in order to define the behaviour of several platelets. The alphabets of the P_i can be included if not implicit.

CSP also provides an *event renaming* operator $P[[R]]$ which can be used to define instances of a generic description. R is a relation between event names, and $P[[R]]$ behaves as P, but the events it performs are renamed through the relation R. For example, if R relates *enter* to *plateletMove*.i.j, then $P[[R]]$ would perform *plateletMove*.i.j whenever P performs *enter*.

In order for a CSP process to interact with a B machine, as shown in Figure 1, we identify input/output communications of CSP processes with machine operations of the B machine, thus treating the operations of a machine as *machine channels*. The process $c?x!v \rightarrow P$ can accept any input x and output v along machine channel c and this matches the B operation `x <-- c(v)`, where the output value v from the CSP description corresponds to the input parameter of the operation, and the input value x corresponds to the output of the operation.

Earlier work [13] has considered restricting the interaction between processes and machines in a CSP || B specification to be communication between a single machine and a single process. The CSP || B communication architecture in this paper is novel because it is more dynamic: communication is permitted between more than one process and a single B machine (see *plateletMove.2.3* in Figure 1). Furthermore, as platelets move along the blood vessel the processes which interact with the B machine change, and so the architecture is more dynamic. For example, when the *Platelet_2* in Figure 1, moves from the second to the third site, the CSP || B architecture changes so that the machine characterising the second platelet communicates with processes $S(3)$ and $S(4)$.

The CSP || B approach supports compositional verification enabling us to focus on the CSP descriptions and B descriptions in isolation. Theoretical results in [13] extend to this architecture, allowing us to make deductions about the model as a whole from the individually verified parts. Thus consistency between the CSP part and the B part can be deduced from consistency of the individual components; deadlock-freedom of the whole model follows from deadlock-freedom of the CSP part; and whether platelets can move, and when they must move, can also be determined from consideration of the CSP aspect of the model.

The benefit of this is that we can apply CSP verification tools and B tools to appropriate parts of the abstract model without ever having to consider the model as one large specification during its verification. Therefore, we can concentrate on verifying properties of the behavioural aspects without having to consider the way in which the B state is updated.

3. Model

We give a very simple model of platelets that move along a line of consecutive positions that represent a vessel. In this model, a platelet can be either smooth or sticky. If it is smooth, it moves freely, restricted only by the fact that it should follow the blood flow and speed. A wound releases chemicals which will act as aggregating agents on platelets; in the presence of sufficient concentrations, a platelet becomes sticky, and releases more chemicals. The concentration of chemicals in a position changes due to diffusion and the blood flow. If two adjacent platelets are sticky, then whenever one of them moves, the other will be dragged behind it. In other words, sticky platelets clump together, and move as a clump. There is no centralised control; a platelet becomes sticky or not, and moves or not, depending on its own state and on that of the platelets and positions in its immediate neighbourhood.

Localised control imposes a challenge: restricting synchronisation to neighbour platelets. Isolation of the different aspects of the problem is also interesting. When exposed to certain chemicals, the state of a platelet changes: it becomes stimulated and sticks to neighbouring sticky platelets, so that their movements affect each other. Therefore, modularisation and separation of concerns require an elaborate protocol, with all the challenges that this entails.

Our solution is a layered architecture to handle separately the movement of platelets, the state of a platelet, and the propagation of chemicals. First, we discuss this architecture; afterwards, we detail the components that constrain the movement of platelets, and those that record their state.

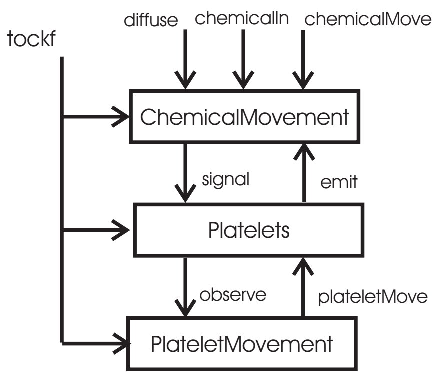

Figure 2. Layered model for platelets

3.1. Architecture

Figure 2 presents the components of our model and the channels that they use to communicate with each other and the environment. A channel *tockf* keeps track of time and determines the direction of the blood flow; the events take the form *tockf.d*, where *d* is a direction. In general terms, in an artery, for example, the flow is continuous in a single direction; in a vein, on the other hand, the blood is pumped. Both kinds of vessels can be simulated using *tockf*.

Several events can take place between two occurrences of a *tockf*; in our model, these events are deemed to occur immediately: during the same instant. An outside observer views the vessel only when a *tockf* occurs.

The component *PlateletMovement* is a partial model of a vessel: a single line of sites through which platelets can move. Another view is provided by the component *ChemicalMovement*: a line of sites that may contain chemicals. Finally, *Platelets* models the state of all platelets.

The channel *plateletMove* is used by a platelet to keep track of its position (site). A site can find out the state of the platelet that it contains using *observe*; this is used to restrict its movements, if it happens to be sticky.

For communication between the components *Platelets* and *ChemicalMovement* we have *signal* and *emit*. The former is used by a chemical site to inform the platelet its amount of chemicals. The latter is used by a stimulated platelet to tell a site the amount of chemicals that it is emitting. Chemicals can also be released, externally, by a wound using the channel *chemicalIn*.

The movement of chemicals is monitored by the component *chemicalMove*. It is different from the movement of platelets, since there are no concerns about clumping. On the other hand, the platelets and the chemicals are both affected by the blood flow determined by *tockf*.

The vessel is represented the parallel composition of processes that model each of these components. The definition of *PlateletMovement* is a parallel composition of processes that model restrictions on the movement of the platelets.

$$\begin{aligned}&\mathit{PlateletMovement} =\\&\quad \mathit{PlatSiteRow} \parallel \mathit{AdjPlatMov} \parallel\\&\quad \mathit{PlatFlow} \parallel \mathit{StickyPlatSite}\end{aligned}$$

- The process *PlatSiteRow* models a row of sites each of which can hold at most one platelet at a time and through which platelets can move at a limited speed: each platelet can move at most once between occurrences of *tockf*.

- The process *AdjPlatMov* restricts the movement of platelets to adjacent sites;

- *PlatFlow* restricts the direction of the movement to that dictated by the last *tockf*.

- Finally, *StickyPlatSite* restricts the movements of sticky platelets: they drag along a neighbouring sticky platelet.

The processes *PlatSiteRow* and *StickyPlatSite* are discussed in the next section.

For the FDR analysis of our model, we use an abstraction of the B machine that defines *Platelets*, and is presented in the next section (Figure 5). The abstraction is an interleaving of a number of platelet sources: processes that represent new smooth platelets, at either end of the vessel. A *Platelet* is modelled as in Figure 3. It has three parameters: its adhesiveness a, its site position i, and a flag that records whether a check of the chemical level at that site has already been done, using the channel *signal*. A platelet can move from i to the positions $i-1$ or $i+1$, as long as it does not leave the vessel. If it is inside the vessel (that is, in a position in the set *Site*), it can be observed, take up a chemical *signal*, *emit* some chemical. When it receives a chemical signal, it nondeterministically chooses to change the state (*PlatNonDetState*); the choice depends of the strength s of the signal, and is only modelled in the B machine. The amount v that a platelet emits is also chosen nondeterministically in the CSP process, and further specified in B (Figure 5). Finally, time is only allowed to pass if the platelet is inside the vessel and has checked the chemical signal or it is not inside the vessel (that is, in a position in the set *Terminal*).

The definition of *ChemicalMovement* is similar to that of *PlateletMovement*; it is discussed in the next section.

$$
\begin{array}{l}
Platelet(a,i,f) = \\
\quad (i > 0)\,\& \\
\quad\quad plateletMove.i.i-1 \rightarrow Platelet(a,i-1,f) \\
\quad \Box\ (i < N)\,\& \\
\quad\quad plateletMove.i.i+1 \rightarrow Platelet(a,i+1,f)) \\
\quad \Box\ i \in SITES\,\& \\
\quad\quad observe.i.a \rightarrow Platelet(a,i,f) \\
\quad \Box\ i \in SITES\,\& \\
\quad\quad signal.i?s \rightarrow PlatNonDetState(i) \\
\quad \Box\ i \in SITES\,\& \\
\quad\quad \bigsqcap v : Value \bullet emit.i!v \rightarrow Platelet(a,i,f) \\
\quad \Box\quad i \in SITES \wedge f == chemicalChecked) \vee \\
\quad\quad\ i \in Terminal\,\& \\
\quad\quad tockf?d \rightarrow Platelet(a,i,chemicalNotChecked)
\end{array}
$$

$$
\begin{array}{l}
PlatNonDetState(i) = \\
\quad Platelet(smooth,i,chemicalChecked) \\
\quad \sqcap Platelet(sticky,i,chemicalChecked)
\end{array}
$$

Figure 3. Abstract CSP model of a *Platelet*

3.2. Components

The top and bottom layers in Figure 2 are each described as a composition of constraints. Constraints on the movement of the platelets and how they interact with each other are captured as behaviour on individual sites. This behaviour is expressed in terms relative to that site and property, using events such as *enter*, *exit*, *pull*, and *drag*. When applied as a component of the whole system, these are mapped (through renaming) into events representing platelet movements between specific sites.

We illustrate the approach by describing some of the constraints on platelet movement through the sites that make up the model. For reasons of space, we restrict the description to include only simplified versions concerned with site capacity, speed limit on platelet movement, and movement of a sticky platelet. We also include a simplified version of the platelet description in B to illustrate how state information is included. The full CSP and B descriptions can be found at `http://www.cs.york.ac.uk/nature/tuna`.

Figure 4 describes the combination of two constraints on platelet movement at a site. Both are concerned with the movement of a platelet into and out of the site, modelled by *enter* and *exit* respectively. The process *PlatCapacity* is concerned with the maximum number of platelets that the site can contain — here this is restricted to just one. The process *PlatSpeed* is concerned with how quickly a platelet can move through the site. This constraint requires consideration of *tockf*: if a platelet enters the site, then it cannot exit

$$
\begin{array}{rcl}
PlatCapacity & = & enter \rightarrow exit \rightarrow PlatCapacity \\
PlatSpeed & = & enter \rightarrow tockf \rightarrow PlatSpeed \\
 & & \Box\ exit \rightarrow PlatSpeed \\
 & & \Box\ tockf \rightarrow PlatSpeed \\
PlatSite & = & PlatCapacity \\
 & & |[\{enter, exit\} \mid \{enter, exit, tockf\}]| \\
 & & PlatSpeed
\end{array}
$$

Figure 4. CSP: Platelet movement at a site

in the same instant. Thus *exit* is blocked until *tockf* occurs. Conversely, a site can allow the *exit* and *entry* of platelets in the same instant provided *exit* is before *entry*, since in this order the events are associated with different platelets.

The combination *PlatSite* of these two constraints describes an aspect of platelet movement through sites.

To combine these constraints, it is necessary to express them in terms of the events of the overall model. For example, *enter* for site $i+1$ (from the left) is the same event as *exit* for site *i* (to the right), and in the combined view both will be considered as $plateletMove.i.i+1$. Combining these views is achieved by use of renaming in CSP, allowing any *plateletMove.j.i* as an entry into site *i*, and any *plateletMove.i.j* as an exit from site *i*, for all $j \in SITES$, where *SITES* is the set of indexes of the sites in the model:

$$
\begin{array}{rcl}
S(i) & = & PlatSite \\
 & & \quad [[enter \longleftarrow plateletMove.j.i, \\
 & & \quad\ \ exit \longleftarrow plateletMove.i.j \mid j \in SITES]]
\end{array}
$$

The combination *PlatSiteRow* is obtained by taking the (alphabetised) parallel combination of these renamed *PlatSite* processes:

$$PlatSiteRow \quad = \quad \Big\|_{i \in SITES} S(i)$$

All processes synchronise on *tockf*, and only $S(i)$ and $S(j)$ synchronise on *plateletMove.i.j*.

Figure 5 gives a B description of the way in which stickiness can be activated and de-activated within a platelet. The variable `level` records the amount of accumulated chemical that activates the platelet; `adh` is the state of the platelet: either `smooth`, or `sticky`; `location` tracks the location of the platelet. This is purely for modelling purposes: platelet behaviour is independent of its location, but the model needs to relate information from the CSP description to the appropriate platelet.

```
MACHINE       Platelet
SETS          ADHESIVENESS={smooth,sticky}
VARIABLES     level, adh, location
INVARIANT     level : NAT
              & adh : ADHESIVENESS
              & location : NAT
INITIALISATION  level := 0 ||
                adh := smooth ||
                location := 0
OPERATIONS
 i <-- plateletMove(j) =
     PRE j : NAT
     THEN i := location ||
          location := j
     END;
 a,i <-- observe =
     BEGIN i := location ||
            a := adh
     END;
 i <-- signal(s) =
     PRE s : NAT1
     THEN level := level + s ||
          i := location ||
          adh := ...
     END;
 s,i <-- emit =
     BEGIN i := location ||
            s := ...
     END;
 tockf = adh := ...
END
```

Figure 5. B model of platelet stimulation

The operations of the machine are equivalent to events in the CSP description, and enable the B machines to synchronise with the CSP physical and chemical layer models using standard CSP synchronisation. For example, *plateletMove.i.j* is a synchronisation between the CSP description of movement between sites, and the B operation `i <-- plateletMove(j)` of the platelet at location `i`. Other operations include `a, i <-- observe`, which indicates the current state of adhesiveness of the platelet; `i <-- signal(s)` which accepts *s* units of chemical (which may activate stickiness, hence the possibility of updating `adh`); `s, i <-- emit`, where the platelet puts chemicals into the site; and `tockf`, which enables changes due to the passage of time (which may deactivate stickiness, updating `adh`). More complex factors influencing stickiness can also be introduced into platelet descriptions by including more complex state, and more sophisticated decision making. This is natural in B, and would be cumbersome in CSP. The facility to handle state-rich parts of the system is the primary motivation for using

$$
\begin{aligned}
\mathit{Stickiness} &= \textit{... check presence and adhesiveness of platelets in current and next site ...behave as NotStuck or Stuck}\\
\mathit{NotStuck} &= \mathit{pull} \rightarrow \mathit{NotStuck}\\
&\quad \Box\ \mathit{drag} \rightarrow \mathit{NotStuck}\\
&\quad \Box\ \mathit{tockf} \rightarrow \mathit{Stickiness}\\
\mathit{Stuck} &= \mathit{pull} \rightarrow \mathit{drag} \rightarrow \mathit{tockf} \rightarrow \mathit{Stickiness}\\
&\quad \Box\ \mathit{tockf} \rightarrow \mathit{Stickiness}
\end{aligned}
$$

Figure 6. Imposing stickiness at a site

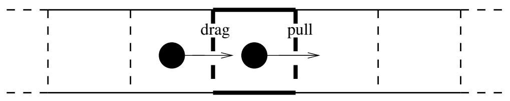

Figure 7. Sticky platelet dragging neighbour

CSP || B over pure CSP to build the models.

When analysing the system as individually verified parts, an abstract CSP description of the `Platelet` B machine is used (Figure 3). It provides the same interface as the B machine in Figure 5: channels *plateletMove*, *observe*, *signal*, *emit*, and *tockf*. The parameters *a* and *i* of the CSP process correspond to the state components `adh` and `location` of the B machine. In CSP, there is no record of the chemical `level`, and in B we do not need the flag *f* to control communication through *signal*.

Figure 6 describes the behaviour resulting from stickiness at a particular site. The behaviour described by *Stuck* is obtained when the site and its neighbour are both occupied by sticky cells. If a platelet exits the site, modelled by occurrence of the event *pull*, then the neighbour must also move, modelled by *drag*, during the same instant. This is illustrated in Figure 7. It is expressed in CSP by the description that *drag* must follow *pull*, before the next *tockf* can occur.

The process *NotStuck* describes the behaviour when there are not two platelets stuck together. In this case, the events *pull* and *drag* can occur independently of each other, and there are no constraints between them.

Each site has a stickiness constraint associated with each direction of movement: leftwards and rightwards. Site *i*'s constraint on rightwards movement, for example, is concerned with the effect of $\mathit{plateletMove}.i.i+1$ (the *pull* event) on $\mathit{plateletMove}.i-1.i$ (the *drag* event).

The stickiness constraints for all the sites are composed into *StickyPlatSite* by renaming and composing in parallel in the same way that *PlatSiteRow* was constructed. A clumping property emerges from this combination of local stickiness behaviours: if a row of sites are all occupied by sticky platelets, then they must move as a clump. When each platelet moves, it must pulls its neighbour in the direction of movement during the same instant, so that they must all have moved by the time of the next *tockf*. In rightwards movement, $plateletMove.i.i+1$ corresponds to *pull* at site i. This must be followed in the same instant by *drag* at site i: the event $plateletMove.i-1.i$. This event is also *pull* at site $i-1$, which will *drag* the next platelet. The effect will ripple along the sites until a site is reached which does not contain a sticky platelet.

The chemical layer also has a process for each of the sites, in this case tracking the local concentration of chemical. Interactions between these sites correspond to the flow of chemical in a variety of ways: in, out, and between sites. Platelets learn the ambient concentration via *signal*, and can increase the concentration via *emit*.

4. Discussion

The work reported in this paper is part of a project to investigate emerging properties in complex systems [14]. In this project, we are interested in novel devices that operate somewhere between the nanoscopic and microscopic levels, and yet cause macroscopic effects. The fundamental research question we are addressing is: can we develop collections of these devices that may be trusted in safety-critical applications? In trying to answer this question, we face the following challenges. (1) To investigate the languages needed for developing abstract models of devices and the protocols needed for inter-device communication and control. (2) To investigate methods for validating and verifying executable models of these devices. (3) To investigate the requirements for an appropriate simulation framework for executable models of devices and of networks of devices. (4) To investigate the nature of safety cases for the use of networks of devices. In this paper, we discuss (1) and (2), while (3) and (4) are discussed elsewhere [14, 17, 11, 16].

4.1. Languages and modelling

Partitioning and structure are fundamental concerns in modelling complex systems, and it is essential to choose an appropriate language and architecture. Given the massive concurrency inherent in modelling blood platelets, a process algebra is a natural choice of language in which to express such models. Using CSP as the process algebra suggests an architecture for the models in which system properties are grouped into layers, with strictly defined interfaces linking them together. This idea is originally due to E. W. Dijkstra, who pointed out the elegant conceptual integrity exhibited by such an organisation [4]. This layering technique allows us to build up very complex behaviours, whilst maintaining a proper separation of concerns. This is a very powerful mechanism: rather than having to model everything in one go, we are able to separate issues of movement of chemicals and platelets, and mechanisms for activating and deactivating stickiness.

Just as there are alternatives to using CSP, there are alternatives to adopting a layered architecture. For example, we could have started from an object-oriented design. This architecture would have had classes for platelets, sites, and chemicals, but then all their behaviours would have to go straight into the relevant class definition. This would have resulted in a very different structure, without the successful separation of concerns that we achieved.

In our models, we aim at a basic principle: all communication should be local. With the exception of the passage of time represented by the clock tocking, we achieve this principle, but this is at the cost of a good deal of intellectual effort. Localised communication is difficult to model in a language based on global synchronisation: it is fragile and it requires very careful handing of event alphabets.

4.2. Validation and verification

The choice of CSP || B has another significant benefit: the availability of the FDR model checker for validating and verifying our models. FDR is a powerful debugging tool, essential when building a complex model. It is particularly useful in getting boundary conditions right, and in negotiating interfaces between layers.

FDR has a built-in check for deadlock, and a simple construction allows us to check for timelocks: behaviour that delays the passage of time. Freedom from deadlocks and timelocks is an internal consistency property of our model. It is a useful guideline for debugging the model, since these inconsistencies generally arise when two local constraints are modelled in incompatible ways.

Other internal consistency properties have been formulated, and FDR allows layers to be checked for these properties before they are composed. For example, we have checked that certain expected patterns of platelet movement are possible for the model. We have also checked the property that once two sticky platelets are adjacent, then they always remain adjacent. As we added more layers, we have gained more confidence in our model. The result is a design that we can use to explore emergent behaviour.

FDR's deadlock checking has also revealed an unexpected emergent behaviour that is an artefact of the model. With only local knowledge, platelets can move independently in either direction. If each end of a clump move in opposite directions, then the middle platelets cannot follow both leaders; yet the stickiness constraints require this before time can pass; hence a timelock results. This tells us that such a model is too flexible: the real world does not behave like this. A solution comes with the notion of flow to dictate the direction of movement.

The process behaviour explorer ProBE allows us to play out different scenarios: we expect certain behaviours to be possible (*e.g.,* a particular pattern of moves), or not to be possible (*e.g.,* time blocked until a clump has all moved). Exploring these behaviours enables us to perform sanity checks with respect to what the model should be doing.

It is difficult to imagine building a model as complex as this without tools like FDR and ProBE, although ProBE is very difficult to use on examples of this scale. The tree structure generated by possible choices is difficult to explore easily, and a more graphical simulation-style interface would make it easier to explore the model. The lack of modules in the version of CSP that we are using makes it awkward to describe our layered architecture, though a simple notion of modules is present in FDR 2.82.

4.3. Applications and experiments

Our work involves studying the feasibility of modelling and analysing large numbers of agents cooperating with only local communication to achieve some significant emergent property: aggregation in the presence of a signal and not in its absence. We have begun to study an architecture for these kinds of models that has a tractable analysis technique. The most obvious application of this architecture is to model the behaviour of other blood cells. Of course, from a physiological viewpoint there are many similarities between the various kinds of blood cells: they all have a common origin in the bone marrow, where they begin as stem cells; only later do they differentiate into red and white cells and platelets. Our models of movement and chemical communication are applicable to all blood cells throughout their life-cycle, and form a common basis on which to build further layers that differentiate their behaviours.

The architecture also serves as a basis for further work to produce ever-more realistic models of platelets. One purpose of these models is to validate components for electronic experiments. Using a simulation platform, we can conduct large-scale experiments to study the emergence of properties such as *haemostasis*. Initial results are presented in a companion paper [17].

These *in silico* experiments can just as well be carried out *in vitro*, or even *in vivo*; however, the electronic experiments are distinguished by starting from a collection of local rules of behaviour. The electronic experiments demonstrate that certain desired behaviours can arise from a specification of these rules. This theory of local behaviour and the results of associated experiments then provide the setting for studying the absence of undesirable behaviours (such as different kinds of thrombosis), the specification of a design for an artificial platelet, and the foundation for its safety case.

A challenge for nanoengineering is to devise a design that satisfies our rules of behaviour.

5. Conclusions

Much of the research work on nanotechnology is related to the physical construction of nanites; the systems engineering aspects have received significantly less attention than fabrication. In this paper we have discussed a proposal to provide techniques to model, analyse, and simulate such systems in order to support the construction of safety cases for such critical mechanisms. We have focused on developing techniques that could be used to build and analyse formal models of complex systems which exhibit emerging properties, and demonstrated them using a model of moving platelets and their associated clumping behaviour in a one-dimensional space. We have also investigated how the model could be extended to deal with platelet movement in a multi-dimensional space, how wounds and constriction of a blood vessel could be described, and the properties that emerge as a consequence of these additional constraints.

Our CSP $\|$ B model scales up to two and three dimensional spaces so that platelets can move around and clump in a richer and more realistic space. We are able to reuse the layering technique presented in the paper but several issues arise: we have to change the notion of platelet movement to include locations that are representative of the appropriate multi-dimensional space, and clarify the constraints which determine how platelets can move. We also have to extend the notion of stickiness of platelets so that they consider all their neighbours when determining whether to become active or not, and whether to return to a smooth state or not.

Formal models have also been used to explore the behaviour of stem cells. D'Inverno and Saunders [3] have developed a multi-agent approach which begins with a formal model, written in Z, and is used as a basis for simulating the behaviour of stem cells. Their aim in developing a mathematical model is to highlight which properties are required of stem cells in order to maintain the body's homeostasis. Their formal models are based on a layering technique in which the physical, chemical and biological environments are considered separately. This separation in a formal model resonates closely with the layering style that we adopted in our models. Our adoption of a layering technique was done in complete isolation from the work of d'Inverno and Saunders but it is clear that in order to deal with the complexity of the problem both approaches consider this layering to be an important factor in the construction of formal models. Through the visualisations in [3] the emergent behaviour of the individual interactions of individual stem cells has been observed. We have also developed visualisation of our formal models using Occam [17] so that we can observe the movement and clumping of platelets amd that the platelets induce haemostasis.

Our future work will establish a link between CSP || B and *Circus* [18, 2], so that the abstract models of a platelet and its protocols may be translated from the former to the latter. The translation will form the starting point for the refinement into an executable model. This will enable us to validate that emergent behaviour that has been identified at the abstract level is what is visualised in a simulation.

Our architecture helps to subordinate the complexity of the model, but its greatest value must lie in its application to other models. When we build other models in this domain, we expect to discover common structures and interconnection strategies, as well as to exploit existing ones.

Acknowledgements

The collaboration that resulted in this piece of work is funded by the EPSRC. We have benefited from extensive discussions with Susan Stepney, Fiona Polack, and Peter Welch, and we are grateful to them, and to Rob Delicata and Heather Turner, for comments. Susanna Hourani provided us with her expertise on haemostasis.

References

[1] J.-R. Abrial. *The B Book—Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] A. L. C. Cavalcanti, A. C. A. Sampaio, and J. C. P. Woodcock. A Refinement Strategy for *Circus*. *Formal Aspects of Computing*, 15(2 - 3):146 — 181, 2003.

[3] M. d'Inverno and R. Saunders. Agent-based modelling of Stem Cell organisation in a Niche. *Engineering Self-Organising Systems : Methodologies and Applications, LNAI*, Springer-Verlag, 2005.

[4] E. W. Dijkstra. The structure of the "THE"-multiprogramming system. *Communications of the ACM* **11**(5):341–346 1968.

[5] K. E. Drexler. *Engines of Creation: the Coming Era of Nanotechnology*. Anchor, 1986.

[6] K. E. Drexler. *Nanosystems: Molecular Machinery, Manufacturing, and Computation*. Wiley, 1992.

[7] R. Freitas. *Nanomedicine*. Landes, 1999.

[8] R. Freitas. Clottocytes: artificial mechanical platelets. *Nanomedicine*. `www.imm.org/Reports/-Rep018.html`, 2000.

[9] M. Goldsmith. *FDR2 User's Manual version 2.80*. Formal Systems (Europe) Ltd., 2003.

[10] Formal Systems (Europe) Ltd. *ProBE User's Manual version 1.30*, 2003.

[11] F. Polack, S. Stepney, H. Turner, P. H. Welch, and F. R. M. Barnes. An Architecture for Modelling Emergence in CA-Like Systems. *Advances in Artificial Life, 8th European Conference on Artificial Life (ECAL 2005)*, LNCS 3630, Springer, 2005.

[12] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.

[13] S. Schneider and H. Treharne: *CSP Theorems for Communicating B Machines*. FACS 17(4), 2005.

[14] S. Stepney and H. Turner and F. A. C. Polack. Engineering Emergence. *ICECCS 2006*, IEEE Press, 2006.

[15] H. Treharne, S. Schneider. Using a Process Algebra to control B operations. *IFM'99*, 1999.

[16] P. H. Welch and F. R. M. Barnes. Mobile Barriers for occam-pi: Semantics, Implementation and Application. *Communicating Process Architectures*, IOS Press, 2005.

[17] P. H. Welch and F. R. M. Barnes and F. A. C. Polack. Communicating Complex Systems. *ICECCS 2006*, IEEE Press, 2006.

[18] J. C. P. Woodcock and A. L. C. Cavalcanti. The Semantics of *Circus*. *ZB 2002*, LNCS 2272, Springer 2002.