Using the CoSMoS Process to Enhance an Executable Model of Auxin Transport Canalisation

Philip Garnett[1], Susan Stepney[2], Francesca Day, and Ottoline Leyser[1]

[1] Area 11, Department of Biology, University of York, YO10 5YW, UK
[2] Department of Computer Science, University of York, YO10 5DD, UK

Abstract. We describe our use of the CoSMoS process to structure an incremental change of a biological simulation. The domain is auxin transport canalisation. An existing simulator is refactored to handle aspects of 2D and 3D space more efficiently, and enhanced to include more realistically-shaped plant cells. The CoSMoS process supports clear separation of concerns, allowing us to concentrate on the biological model and the implementation decisions separately. This gives a clear and well-justified simulator design that can be validated by biologists, yet still allows efficient implementation.

1 Introduction

Biological systems present many challenges to science, particularly due to the complex nature of biology itself. Many biological processes are highly connected, making it hard to study them in isolation. It is frequently difficult to get good quantitative data; even good data might lack part of the larger picture. These factors and many others make it difficult to form good assumptions about how a biological process is being regulated; our solutions might reflect our lack of knowledge, rather than offer insight into the real process.

Increasingly biology is looking to modelling to help progress understanding. Developing a simulation of a biological process is a challenging task in itself, but doing so can assist with some of the problems. The modelling process requires the builders to go systematically through the information and data about a system, ideally with experts in the field. Simply going through this modelling process can highlight new areas of focus, or problems and gaps in understanding. The resulting simulation and models can also be a tool for the generation and testing of hypotheses, hiding some of the complexity of the real system but capturing enough to allow the study of the process of interest.

The level of abstraction in a model is critical. Too high, and we risk ruling out the possibility that simulations will produce interesting emergent behaviours that are observed in the real system. Too low, and the simulations produced could be difficult to work with, understand and validate. These factors make the design decisions made when producing a simulation important, as they determine the balance between these conflicting requirements.

A simulation must be developed using a rigorous process of design, implementation, and validation if it is to be scientifically respectable. Additionally, a useful simulation will need to be upgraded and enhanced in a principled manner as its requirements change to address new research questions. The CoSMoS (Complex Systems Modelling and Simulation) process [1] provides a flexible approach designed to support the modelling and analysis of complex systems, including the design and validation of appropriate computer simulations.

We have previously used the CoSMoS process to guide the initial development of a simulation of an abstract tissue level model of plant cells [14]. Here we present work using the same process to guide modification and enhancement of this existing system, by improving the model of space, and allowing more naturally shaped cells. This work helps demonstrate how the CoSMoS process can be used in an incremental manner.

In §2.1 we overview the CoSMoS process as used for modelling, designing, and implementing biological system simulations. In §2.2 we discuss the use of UML as a suitable modelling language to support this process. In §2.3 we give an overview of the initial auxin model. We then use the CoSMoS process components to structure the remaining sections. In §3 we introduce the Research Context. In §4 we summarise the biological Domain Model. In §5 we discuss the issues relating to modelling space that we are addressing in this increment. In §6 we discuss how the Platform Model has been updated using the CoSMoS process. In §7 we conclude with a discussion of our experiences.

2 Background

2.1 CoSMoS Process: The modelling lifecycle

Described in detail by Andrews et al. [1], and used in our earlier work [14], the CoSMoS process provides a systematic approach to building models and simulations of complex systems, including the biological system of interest here. The CoSMoS process does not include a defined end point: the process is incremental, aimed at supporting a series of simulations. We [14] and others [34] have successfully used this process to assist in the production of simulations of complex biological systems. Summarised in figure 1, the process contains the following components (summarised from [1, 14]):

Research Context : the overall scientific research context. This includes the motivation for doing the research, the questions to be addressed, and the requirements for success.

Domain Model : conceptual "top-down" model of the real world system to be simulated. The domain model is developed in conjunction with the domain experts, with its scope determined by the Research Context. The model may explicitly include various emergent properties of the system.

Platform Model : (called the Software Model in [14]) a "bottom up" model of how the real world system is to be cast into a simulation. This includes: the system boundary, what parts of the the Domain Model are being simulated;

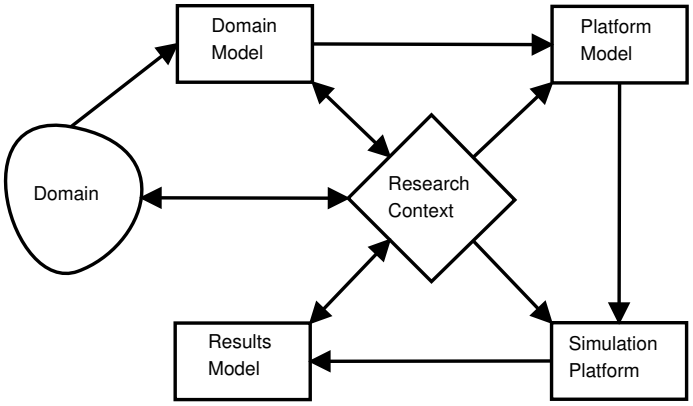

Fig. 1. The components of the CoSMoS process [1, fig.2.1]. Arrows indicate the main information flows during the development of the different components. There is no prescribed route through the process.

simplifying assumptions or abstractions; assumptions made due to lack of information from the domain experts; removal of emergent properties (properties that should be consequences of the simulation, rather than explicitly implemented in it).

Simulation Platform : the executable implementation. The development of the Simulator from the Platform Model is a standard software engineering process.

Results Model : a "top down" conceptual model of the simulated world. This model is compared with the Domain Model in order to test various hypotheses. (This part of our research is beyond the scope of this paper.)

2.2 Modelling biology and simulations with UML

UML (Unified Modelling Language) [27] is a suite of diagramming notations designed to aid in the development of large object-oriented software engineering projects by groups of developers working in teams.

Although UML is normally used in conjunction with an object-oriented programming language, it is well suited to agent-based modelling [26], where an agent can be thought of as an object with its own thread of control, allowing highly parallel systems of multiple agents. Biological 'agents', such as cells and proteins, can be modelled as UML agents. This relatively natural mapping between biological agents and their UML counterparts means that much of the structure of a biological simulation can be well-represented by UML. There are a number of published cases where UML has been successfully used to assist the production of biological models [10, 14, 16, 34, 44].

We have found that UML diagrams (in conjunction with traditional biological 'cartoons') are relatively accessible to biologists, allowing these domain experts

to provide input to the model of the simulation without the need to understand the implementation details.

2.3 Auxin transport canalisation model

Auxin was one of the first plant hormones to be discovered, 130 years ago by Charles and Francis Darwin [9]. Understanding auxin's functions still presents many challenges to plant science as it is involved in diverse aspects of plant patterning and development. Computational modelling plays an important role in auxin transport research [13].

We are using a UML-based approach within the CoSMoS process to design and build a simulation of auxin transport canalisation in the plant *Arabidopsis*. Our initial model and simulation is described in [14]. This approach allows us to build models containing the biological objects that we believe to be involved in auxin canalisation, and then produce simulations to test various hypotheses about the biological processes of interest. If an hypothesis is correct we should see the correct emergent behaviour when the simulation is run; if not we can then return to the UML models and implement our next hypothesis. If all our hypotheses fail to produce the emergent behaviour of interest we might have to return to a different part of the process.

In this paper we describe an enhancement to our initial model in [14]. The most significant modifications are in the Platform Model and associated Simulation Platform. We revisit significant assumptions about what should be removed from the domain model. The main progress made in the design and implementation of our models has been with the handling of the simulation space, allowing the cells of the tissue modelled to be more naturally shaped. The improved 2D simulator has been adapted into 3D.

3 The Research Context

The auxin transport community studies many different aspects of auxin transport. These include, but are not limited to: auxin transport canalisation [37, 38]; shoot branching regulation [21, 22, 28]; leaf venation [41]; and phyllotactic patterning [20, 35]. These processes are concerned with the developmental patterning of a plant, at both the tissue level and that of the whole plant.

Our research sits within this wider community; it uses background biology derived from the literature, and from wet-lab experiments carried out in the Leyser group (for more information see §4). We primarily focus on modelling the process of auxin transport canalisation, within the context of shoot branching regulation.

There are many published mathematical models of auxin transport. We have chosen to develop executable models as we believe this modelling technique lends itself to biological systems, and can offer an alternative perspective [11, 13], particularly as we are modelling the PIN protein transporters at a reasonable level of detail.

Our models focus on the question of PIN cycling and its role in canalisation, and we aim to test different regulatory mechanisms of PIN cycling.

4 The Domain Model: auxin transport canalisation

The domain of our model remains auxin transport. §4.1 summarises the background biology used as input to the Domain Model. The full model is informed by more detailed biological information than is summarised here; we direct interested readers to reviews of auxin transport [4, 3, 7]. §4.2 summarises the way this biology is captured in UML diagrams.

4.1 The Biological Domain

We are developing models to investigate the formation of auxin transport canals in plant tissues. This process of canalisation and its regulation are not fully understood.

Canalisation can be thought of as a self-organising process, where auxin in cells promotes its own transport between cells through the tissue of the plant [37]. In canalisation the transport goes from a source, an area where auxin is accumulating, to a sink elsewhere in the tissue. The link that forms between these two sites is called an auxin canal, and the process by which it forms is canalisation [23, 24, 39]. The transport of auxin between cells is dependent on membrane localised transport proteins, of which the ABCB and PIN transporters are two prominent families [2, 12, 15, 31, 43, 45]. We are primarily interested in the PIN family of transporters. PIN proteins are often distributed asymmetrically around the membrane of a cell. This asymmetry enables directional auxin transport, which is central to canalisation.

We are particularly interested in canalisation within the context of shoot branching regulation. Shoot branching is the process where lateral axillary buds on the main stem of a plant activate and grow into branches [22]. Auxin produced higher up the plant inhibits the growth of lateral axillary buds, a phenomena known as apical dominance [8]. If the auxin sources inhibiting a bud are removed by decapitating the plant the bud is released and is able to grow. This can be reversed by application of auxin directly to the site of decapitation. We believe that the bud is able to grow only when it can export its auxin into the main stem.

The vascular link between an active growing bud and the main vascular tissue in the stem requires auxin transport canalisation from the bud to the stem to trigger its differentiation. It is this canalisation process we would ultimately like to model, as understanding canalisation at this position in the plant could help with the understanding of shoot branching. In order for an auxin transport canal to form between the bud and the main stem, the stem vasculature must behave as a relatively strong sink when compared with the surrounding tissue. If the stem vasculature is already transporting large amounts of auxin from higher up the plant, its sink strength is reduced, the canal does not form, and the bud does

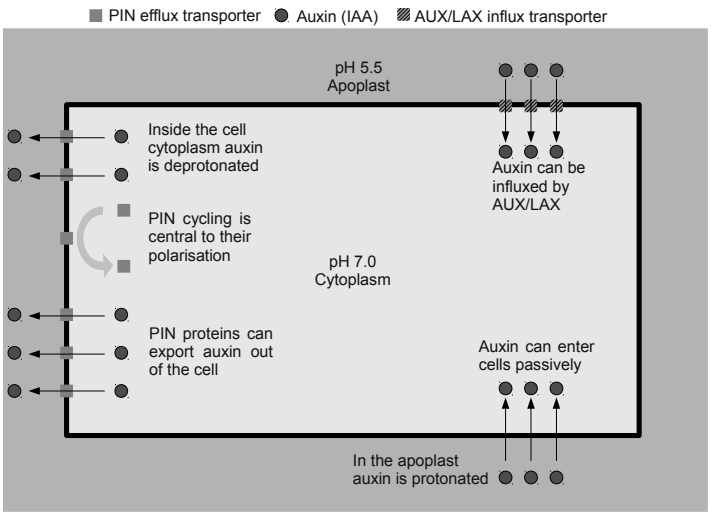

Fig. 2. Domain Model PIN Localisation: Auxin transport into and out of cells is central to canalisation. Protonated auxin in the apoplast is able to enter the cell passively, or to be actively influxed by AUX/LAX transporters. Once inside the cell the majority of auxin is deprotonated and is therefore unable to leave the cell unaided. This is often known as the Acid Trap hypotheses [36, 33]. PIN transports are important to the efflux of auxin from cells. The regulated cycling of PINs on and off the cell membrane causes them to become localised asymmetrically around the cell membrane. This process is not fully understood, but is critical to the directional transport of auxin in tissues, and the process of canalisation.

not activate and is unable grow into branch. However, if the the level of auxin in the stem starts to fall, its sink strength increases; this allows canalisation to occur and a canal is able to form. Auxin can be exported out of the bud, causing it to activate and grow into a new branch.

This process of bud activation has been successfully modelled mathematically on a tissue and whole plant scale [32]. However, there are processes occurring at the cellular level that are not fully understood. Auxin has an interesting cell biology that is responsible for some aspects of its behaviour (figure 2). Auxin is a weak acid and therefore some auxin is able to enter cells passively from the more acidic apoplast (intercellular space) by crossing the cell membrane. It can also be actively transported into cells by AUX/LAX influx carrier proteins [30]. Once in the pH-neutral cytoplasm the majority of auxin is deprotonated, and therefore unable to recross the membrane passively. It is essentially trapped, a phenomena known as the Acid Trap hypothesis [33, 36]. Auxin is only able to leave the cell via efflux transport proteins. We are interested in the PIN family of transporters, as they are found to be polarly localised in cells that form auxin transport canals and are therefore very likely to be central to canalisation [40].

The process of canalisation has been the focus of much prior work over a long period. Sachs suggested a model where auxin facilitates its own flow: both the ability of a cell to transport auxin and the polarity of the auxin flow increase with the amount of auxin being transported [39]. Therefore as the transport capacity increases the cells in the canal become better sinks and draw in more auxin from their neighbours. Mitchison modelled this process mathematically and was able to show it to work [23, 25]. Mitchison's models predict canals of high flow and low concentration, where as experimental evidence suggests that there is both high flux and *high* concentration [5, 42]. Kramer later produced models that showed that the addition of the AUX/LAX auxin influx proteins can allow for canals of both high flux and high concentration [17].

We now have more information about the biology of canalisation. Experiments show that auxin is up-regulating its own transport by increasing the amount of PIN protein available to transport auxin [29]. Thus the more auxin in a cell, the more it can transport. This has been further confirmed by experiments showing that if the negative regulators of PIN accumulation are removed, auxin transport increases and the stem is able to transport more auxin [6]. The other key part of the process is the localisation of PIN to provide the directional transport of auxin. However, the mechanism of PIN localisation is not understood.

PIN proteins are therefore of great interest to the canalisation process as they export auxin out of the cells, and their polar localisation patterns are responsible for complex transport patterns in a number of plant tissues [18, 19]. However, what directs the PIN in the cells into the observed polar patterns remains an important question: if PIN is positioned by detection of auxin flux, as Sachs suggests [39], what is it in cells that is detecting auxin flux? This is one problem our simulations aim to address.

4.2 Domain Model UML

The UML used to capture the Domain Model has not changed significantly during the development process. We briefly summarise it here, but direct interested readers to our previous paper for more detailed discussion [14].

Domain Model use cases. These capture a high level view of what the system does, such as the regulation of proteins and hormones.

Domain Model class diagram. This captures the biological entities of interest as objects and classes. Objects map naturally to biological entities such as proteins, hormones, and cells. Cells themselves are composed of a number of objects such as membranes, cytoplasm and vacuoles, which are associated with each other in space. We also need to regulate the production of agents like proteins and hormones, which is done by cells. See figure 3.

Domain Model state diagrams. These are among the most useful of the Domain Model diagrams for communicating with the Domain Experts, as they

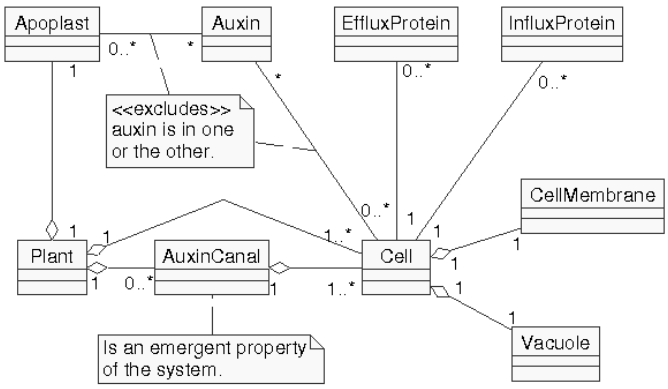

Fig. 3. Domain Model class diagram [14, fig.4].

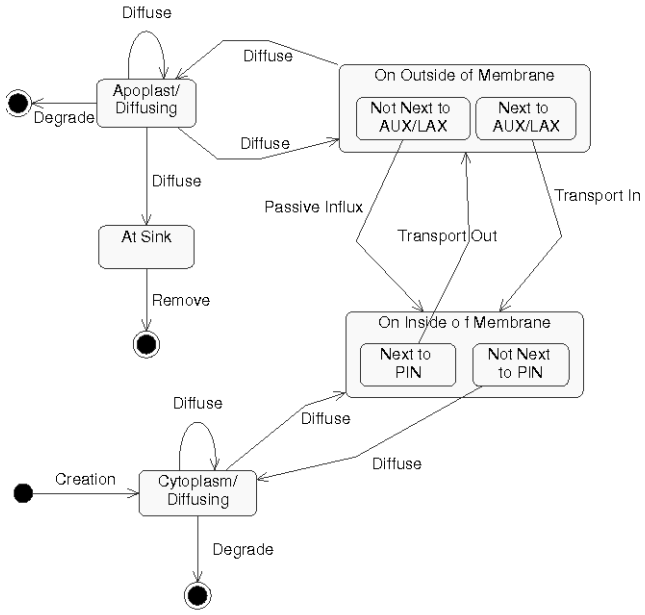

Fig. 4. Domain Model auxin state diagram [14, fig.6].

appear to map well to the way these biological processes are understood. State diagrams capture how an object changes through time. They are able to show the different possible states of the biological objects, and how an object moves from one state to another. Some spatial information can also be captured by state diagrams, as the changes can be associated with a location, within and outside a cell. For example the possible state changes that the auxin object can undergo are different depending on whether it is inside or outside a cell. State diagrams map neatly to the traditional biological 'cartoon' showing process occurring in cells (such as figure 2). The behaviour of auxin can be cross-referenced between the 'cartoon' and the Domain Model auxin state diagram (figure 4).

5 Modelling Space

Here we discuss an important part of the model that was not explicitly dealt with in the first increment [14]: space.

The simulation space is part of the biological domain that cannot easily be captured using UML, and might be based on assumptions that could escape recording. The space in which our biological entities exist is implied in the UML. We can see from the domain class diagram (figure 3) that we are representing part of a Plant built from a number of Cells (each with a CellMembrane and Vacuole), surrounded by Apoplast. However the nature of the space is not captured, nor is any information about how the objects such as CellMembranes or Vacuoles are arranged into Cells, nor how the Cells and Apoplast are arranged into a plant tissue. This information might seem obvious, since it is easy to imagine (particularly if you work in the field of plant science) what a small 2D section of plant tissue might look like. This aspect is easy to capture with a more traditional 'cartoon' and explanatory documentation.

In our initial simulation the assumption is made that a 2D rectangular 'box' is an adequate representation for a plant cell. Therefore the initial simulation is limited to 2D cells of four straight sides. This is a reasonable simplification to make; mature cells in the stem of a plant are often fairly block-like in shape. However, auxin transport canals also form through tissues with cells of varying size and shape, particularly at the interface of a bud and existing vascular tissue. Therefore being able to test the behaviour of our hypothesised regulation of PIN localisation in cells of more natural shapes would be interesting both from a biological and simulation point of view.

Linked to this is the need to try to investigate the effect that 3D cells would have on the behaviour of the hypotheses. There are a number of differences between real 3D cells and simulated 2D cells that might have an effect on the PIN localisation. Being able to simulate even a small number of 3D cells could provide interesting insight into the effect of abstracting 3D cells into 2D. Early simulations have been done in 3D, but it is not well implemented in the initial simulation. We also want to allow for more naturally shaped 3D cells. The first of these issues are linked to the way in which space (the environment of the agents) in the model is handled. This impacts a number of key areas: the interaction

between the agents and the space, and how the space is split up into cells and the other structures in the plant tissue.

These modifications are more about changes in the level of abstraction assumed during the development of the Platform Model, about how the simulation is to be constructed from the Domain Model. Sometimes it is possible to change existing simulation code to allow for the change in abstraction. In our case the changes are significant, and the development process of the first simulation highlights a number of areas where improvements could be made.

6 Platform Model

The Platform Model includes all the extra components that allow the simulation to run. This includes all the processes required to get the simulation to a point where it is able to start, such as generating the space and populating it with cells.

The Platform Model has three kinds of information: biological processes captured directly from the Domain Model; biological processes required for the proper functioning of the simulation, but not of explicit interest to the Research Context, implemented with regard to efficiency rather than biological fidelity; instrumentation and other such aspects of a simulation that are not part of the Domain, but are needed to observe and document the simulation results.

Throughout the continued developmental process it is the Platform Model that has seen the most change. Not only have we made efforts to make the simulated space more realistic with respect to the real plant, but huge improvements have been made in the data output from the simulations and the organisation of the code.

6.1 Platform Model UML

Platform Model use cases: these capture the user requirements for using the simulator, the traditional use for use cases in software engineering. These are unchanged from the original version [14].

Platform Model class diagram. This is produced from the Domain Model class diagram, with all emergent properties (such as the Auxin Canal) removed. This high level diagram shows mainly the biologically relevant parts of the model, and is unchanged in this iteration (figure 5).

Platform Model class diagram, implementation level. As we move towards code, implementation level data structures are added to the class diagram. §6.2 discusses the changes to the implementation level Platform Model class diagram.

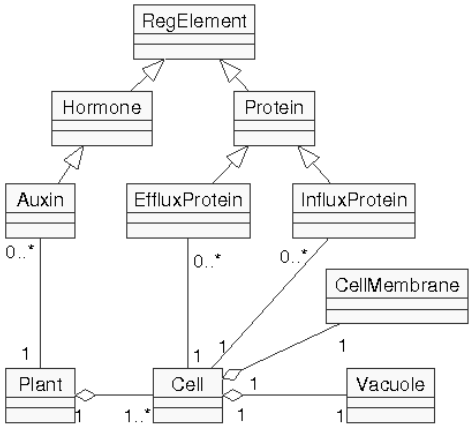

Fig. 5. Platform Model class diagram [14, fig.10]. Note that space is not explicitly dealt with, rather it is generic unless something like a CellMembrane object is put into a position.

Platform Model state diagrams. These follow the Domain Model state diagrams and remain largely unchanged from the original version [14].

As the simulator increases in complexity, keeping the high level and implementation level Platform Models distinct becomes increasingly important. Things that are not biologically relevant, but are needed in a simulator, such as the ability to easily checkpoint to allow restarting, add complexity to the model that biologists do not need to see. We therefore omit such detail from the high level Platform Model diagrams discussed with the biologists, and retain it in implementation level Platform Model diagrams used by the developer.

6.2 The Division of Space

The main changes we made in moving from the initial to the enhanced version were to the way the space is handled in the Platform Model and simulation.

The initial version treats the space as a largely homogeneous area, a grid of pixels, on which cell membranes and vacuoles are drawn, dividing the space into separate areas. Some areas are associated with objects like Vacuole and CellMembrane; other areas are essentially null.

A CellMembrane is a continuous line enclosing the cell (figure 6A). It is straightforward to define a cell membrane if it is built from straight line segments. However it is more difficult to define realistic-shaped cells with curved membranes (figure 6B) using this approach. The membranes would need to be drawn correctly somehow, and then read into the simulation. It would be easier to place the cells into the space as continuous areas of cytoplasm, and then determine the position of the membranes around the edge (which is how it is implemented in the enhanced version). A new method of handling the space

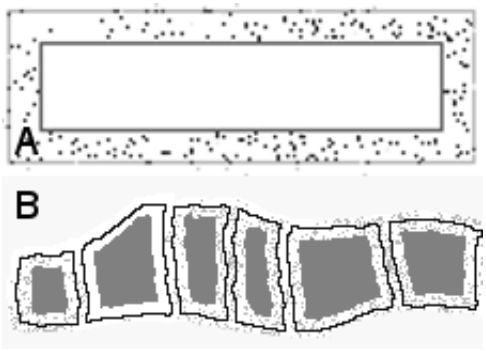

Fig. 6. (A): Section of visual output from the initial simulator. The thin line of the cell membrane (outer grey line) is drawn into the space to define the cell. The vacuole is defined by drawing another membrane (darker grey line). This is a simple task for boxes, but more difficult for natural shapes. (B): Section of visual output from the enhanced simulator, showing a continuous curved membrane (black line).

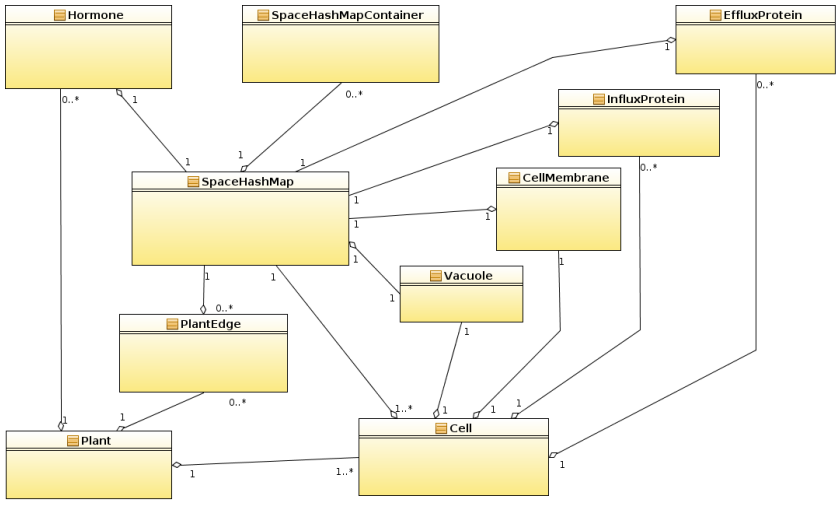

Fig. 7. Implementation level Platform Model class diagram of the initial simulator. All objects in space require access to a singleton class SpaceHashMap that provides them with information about the space they are in, via the SpaceHashMapContainer class. As more kinds of space are needed, the resulting code becomes inefficient and untidy. (Inheritance has been left off this diagram to improve readability.)

needs to be able to address such issues. We also want it to be easier to extend the range of different types of space that could exist in the simulation.

In the initial version of the model, all space is described by a single object. Figure 7 shows the relevant part of the implementation level Platform Model class diagram. A single class, SpaceHashMapContainer, has different attributes that allow it to represent all of the different types of space in the simulation, depending on the values the attributes are given. However, the complexity and size of this class increases each time we add a new kind of area of space in the simulation.

Another significant issue with having all the kinds of space specified in a single class is that some of the methods in the class need to behave differently depending on what the kind of space is. This increases the complexity of the individual methods in the class. The organisation of the code also suffers from having added the space to the model, rather than it having been designed with space in mind.

For the enhanced version, we refactor the code to handle the space in a more area-specific manner, to improve its structure and extensibility, and to allow more natural-shaped cells.

In the initial model, space is general unless it is given a particular type. In the enhanced model, all the space is given an area type. An abstract class Area has attributes common to all the different types of area in the simulation. Sub-classes extend the abstract Area class into more specific kinds of space. Currently there are five types of area. Cells have Cytoplasm, Membrane, and Vacuole. Outside the cells there is Apoplast: the cell walls. Finally there is EmptySpace; this is used to allow more elaborate shapes of space to be used in the models, and is not processed. Apoplast areas separate all the cells from each other, and also separate cells from EmptySpace. See figure 8.

The abstract Area class contains many attributes and methods common to all the different types of area. These attributes and methods tend to be the system aspects of the class, such as accessing the colour of the object or its position in the space. The specific area type then adds extra methods that give that space more biologically specific behaviour, and if necessary overload particular methods. This has many advantages, including simplicity of code maintenance reducing the likelihood of introducing errors. When a new type of space is added to the model much of the code is already in place.

6.3 Agents in Space

In the initial version, the code that determines how the agents move around in the simulation space is held in the agents themselves. This results in the classes describing the agents becoming more complicated each time a new kind of space is added to the simulation. The agent requests information about its current environment from the environment directly. It then uses this to make an appropriate decision about what it would do. There is also an inconsistency in where the agents are stored. Figure 7 shows that auxin (Hormone objects) are held in the Plant class, but the proteins are in the Cell class. This makes

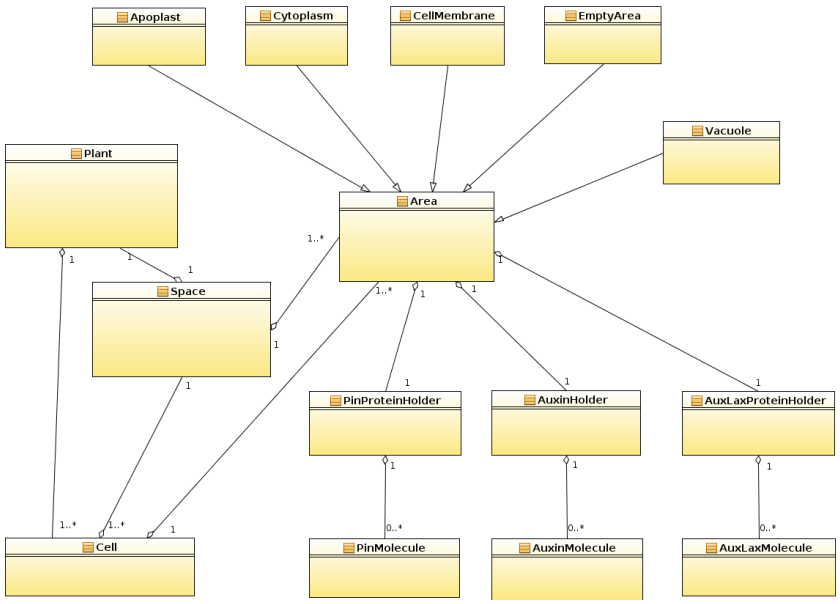

Fig. 8. Implementation level Platform Model class diagram of the enhanced simulator. The space is now built from different child classes of the Area class, each with a holder looking after the different Molecules. The Space contains many areas which compose a single Plant, the Plant has many Cells. Cell requires access to the Space directly but also contain within it a list of all its associated Areas. A Cell does not directly contain any Molecules. (Inheritance of the different Holder and Molecule classes are not shown, to improve diagram readability.)

biological sense, since the PIN and AUX/LAX proteins do not leave the cell, but auxin does. However it makes better *implementation* sense to think as all three as being held in the Space, and whether or not this is in a Cell is determined by what the space is. This is the case for the enhanced simulator, as shown in figure 8.

The movement of agents is also the responsibility of the Space in the enhanced simulator. Each Area sub-class that can have agents contains an AgentHolder with methods for storing the agents that are contained within it. The different AgentHolder sub-classes (such as AuxinHolder) for each agent inherit properties from the parent AgentHolder, but are also given specific behaviours. The AgentHolder classes accept incoming agents to their area. The movement of the agents is controlled by the Area sub-class, which has methods for moving any agents in the relevant AgentHolder. This puts the responsibility of moving agents onto the Area class. Therefore when a new kind of space is added, the areas are updated to allow agents to move into this new kind of space. These changes are reflected in figure 8 (the inheritance from the abstract AgentHolder class and the Molecule class are not shown, to improve the readability of the diagram.)

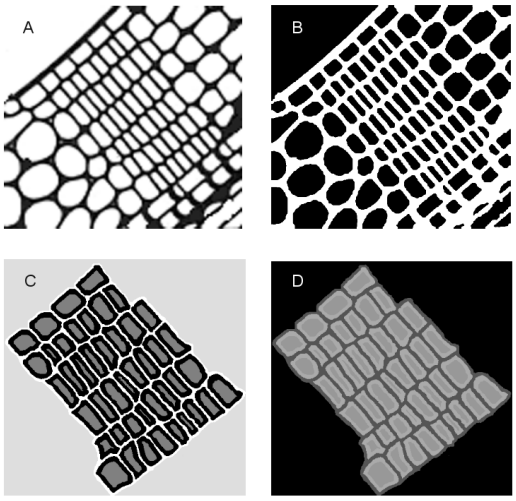

Fig. 9. Processing sections of plants into the model. If the section photos are of high enough quality the processing can be done automatically. (A) Photographic section from a real plant, tided up to allow it to be processed. (B) Image processed for reading into the model: black areas will become Cytoplasm, white areas Apoplast. (C) Image modified by hand to isolate a patch of cells: light grey areas will become EmptySpace. Vacuole areas are then added automatically (dark grey). (D) Template as it finally appears in the simulation visualisation. CellMembrane areas are added automatically at the interface between Cytoplasm (here light grey) and Apoplast (here dark grey).

6.4 Space from Templates

The more natural-shaped cells are defined using templates derived from images of real plants.

Figure 9 shows the lifecycle of a template: it starts as an image of a section of a plant, and ends as a representation of the simulation space. Templates can either be generated automatically (normally with a little manual processing), or fully by hand. They need to contain only three pieces of information: the areas of the space that are empty (not active as simulation space but required to be spatially present); the areas that are apoplast; and the the areas that are cells. The template is then processed to add vacuoles into the cells. These are not added directly from the image being used, because simulated 2D cells need smaller vacuoles than are shown in sections of real 3D cells. Instead they are added automatically by filling the centre of the cell a certain amount (see §6.5 for discussion of this). Cell membranes are also added automatically around the cytoplasm. Once the vacuoles and cell membranes have been added into the space we essentially have areas presenting cell cytoplasm, cell membranes, vacuoles, apoplast and any empty areas. All are displayed as different colours in the image (shown as different shades of grey in the figure).

In the simulation the space is created to match the pixel size of the template, and the entire space starts off as apoplast. Each pixel of the template is then read and its colour determines what it is in the space. The next task is to group areas of continuous cytoplasm and the vacuole inside them into the more abstract notion of a cell. In a plant, a cell is essentially a container of elements that need to be held together. The elements have no concept of togetherness, they are just associated in space. The way the different elements interact is through the common environment. In the simulation a cell is more abstract. It is similar in that it contains lists of all of its spatial contents but it also needs methods to create more proteins or hormones when they are required. Essentially the nucleus of a real cell, which regulates what is expressed, is part of the more abstract Cell class in the simulation. The Cell class provides access to the common environment, to allow cell regulation.

6.5 3D Space

Our initial simulator version can handle 3D models, but not very efficiently. The enhanced simulator space is implemented by ensuring that all Areas know who their neighbours are, and therefore the move to 3D is much simpler as it mainly involves giving the Areas more neighbours. The code for the 2D and 3D versions of the simulator are therefore very similar, which makes it much easier to maintain.

We can either generate block-shaped 3D cells from algorithms, or naturally shaped cells by stacking prepared 2D templates together in a careful order to create a 3D space. This requires three kinds of templates, containing: only Apoplast; Apoplast and Cytoplasm; Apoplast, Cytoplasm and Vacuole.

We are interested in 3D simulations to investigate how our hypotheses behave in 3D, and the effect of using 2D simulation, particularly on the effects of vacuoles. Compare the possible paths an auxin molecule can take in a 3D cell with a large vacuole to that of a 2D cell with a large vacuole. We can see from figure 10 that in a 3D cell taking the path through the vertical section is much longer than taking a path through the horizontal section at roughly the position of the dashed line. In the 2D cell there is only the vertical path. All other diffusing agents will have the same problem. This could have an effect on auxin transport in a 2D tissue. We can use the 3D simulation to help calibrate the required size of the 2D vacuole.

7 Discussion

We have used the CoSMoS process to produce an incremental change to a pre-existing CoSMoS-based model and simulator. The enhanced simulator has improved performance, allowing us to run simulations of canalisation over larger arrays of cells, and over more naturally-shaped cells. Canals still form in the latter case, indicating that the observed process is not an artefact of the rectangular cells. The biologically-relevant results from this enhanced simulator version will

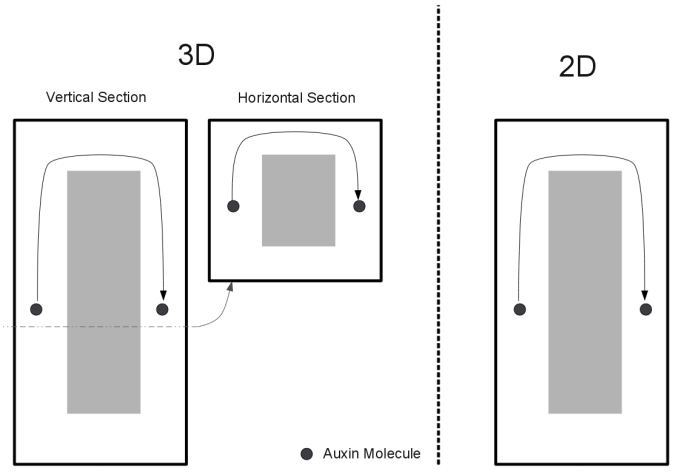

Fig. 10. Comparison of possible paths of auxin molecules (or other agents) in 2D or 3D cells. In the 3D cell the auxin has the possibility of taking a short path to the same position. This is not possible in a 2D cell with only one path.

be presented elsewhere; here we discuss the impact of the CoSMoS process on the development.

Continuing to develop our simulations with the CoSMoS process assisted by UML has allowed us to progress in an efficient and systematic way. Using this approach helps us to identify which of the assumptions we made when making the transition to the Platform Model from the Domain Model might need to be reassessed. Both the CoSMoS process and UML have allowed us to see how progressing down a particular development path was increasing the gap between the biology we were trying simulate and how we were implementing it.

The CoSMoS process ensures that at each stage of modelling and simulator development effort is made to understand and acknowledge what decisions have been made and why. It is also flexible enough to work with software engineering tools like UML. UML is able to produce detailed information about the structure of a biological system. It is then possible to extend these UML descriptions of the biology into code skeletons of a simulator, even though the final UML and code include much more than just the underlying biology. That underlying structure should be visible (visibility can be improved by maintaining a separate Platform Model and Refined Platform Model), and areas where it has had to change or has been deliberately changed (such as the removal of emergent properties) can be highlighted and the reasons made clear. UML diagrams, particularly state diagrams, can be compared with more traditional biological 'cartoons' to enhance cross-disciplinary communication of model structure and included biology. This can help increase information flow between modellers and domain experts.

Going through the CoSMoS process has allowed us to see that we needed to return to the Platform Model of our simulator to include more natural cell

shapes derived from the biology. Both the CoSMoS process and UML allowed us to identify parts of the simulator code that were becoming over complicated and could be improved. From this we were able to improve how the biology of the Domain Model is captured in the Platform Model, and simultaneously improve the simulator code itself.

In the future we may wish to include more aspects of the Domain in the Models and simulation. One important example is growth. Introducing growth into the current simulation architecture would be very difficult to do. Therefore the CoSMoS process could be used to make the transition between the current simulator to a new one in a way that allows us to fully understand the differences between the two simulators produced.

Acknowledgements

This work is supported by a BBSRC/Microsoft Research CASE studentship and an EPSRC TRANSIT project (EP/F032749/1) summer studentship. Thanks to Lauren Shipley for comments on an earlier draft of the paper, and to the anonymous referees whose suggestions have helped improve the clarity of this final version.

References

1. Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, 2010.
2. A. Bandyopadhyay, J. J. Blakeslee, O. R. Lee, J. Mravec, M. Sauer, B. Titapiwatanakun, S. N. Makam, R. Bouchard, M. Geisler, E. Martinoia, J. Friml, W. A. Peer, and A. S. Murphy. Interactions of PIN and PGP auxin transport mechanisms. *Biochem. Soc. Trans.*, 35(Pt 1):137–141, 2007.
3. René Benjamins, Nenad Malenica, and Christian Luschnig. Regulating the regulator: the control of auxin transport. *Bioessays*, 27(12):1246–1255, 2005.
4. René Benjamins and Ben Scheres. Auxin: the looping star in plant development. *Ann. Rev. Plant Biol.*, 59:443–465, 2008.
5. Eva Benková, Marta Michniewicz, Michael Sauer, Thomas Teichmann, Daniela Seifertová, Gerd Jürgens, and Jiří Friml. Local, efflux-dependent auxin gradients as a common module for plant organ formation. *Cell*, 115(5):591–602, 2003.
6. Tom Bennett, Tobias Sieberer, Barbara Willett, Jon Booker, Christian Luschnig, and Ottoline Leyser. The arabidopsis MAX pathway controls shoot branching by regulating auxin transport. *Curr. Biol.*, 16(6):553–563, 2006.
7. Yohann Boutté, Yoshihisa Ikeda, and Markus Grebe. Mechanisms of auxin-dependent cell and tissue polarity. *Curr. Opin. Plant Biol.*, 10(6):616–623, 2007.
8. Morris Cline. Apical dominance. *The Botanical Review*, 57(4):318–358, 1991.
9. Charles Darwin and Francis Darwin. *The Power of Movement in Plants*. John Murray, 1880.
10. S. Efroni, D. Harel, and I. R. Cohen. Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. *Genome Res.*, 13(11):2485–97, 2003.

11. Jasmin Fisher and Thomas A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007.
12. Jiří Friml, Anne Vieten, Michael Sauer, Dolf Weijers, Heinz Schwarz, Thorsten Hamann, Remko Offringa, and Gerd Jürgens. Efflux-dependent auxin gradients establish the apical-basal axis of arabidopsis. *Nature*, 426(6963):147–153, 2003.
13. Philip Garnett, Arno Steinacher, Susan Stepney, Richard Clayton, and Ottoline Leyser. Computer simulation: the imaginary friend of auxin transport biology. *BioEssays*, 32(9), September 2010.
14. Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an executable model of auxin transport canalisation. In *CoSMoS 2008, York, UK, September 2008*, pages 63–91. Luniver Press, 2008.
15. Markus Geisler and Angus S. Murphy. The ABC of auxin transport: the role of p-glycoproteins in plant development. *FEBS Lett.*, 580(4):1094–1102, 2006.
16. Na'aman Kam, Irun R. Cohen, and David Harel. The immune system as a reactive system: Modeling T cell activation with statecharts. In *HCC '01*, page 15. IEEE, 2001.
17. Eric M. Kramer. PIN and AUX/LAX proteins: their role in auxin accumulation. *Trends Plant Sci.*, 9(12):578–582, 2004.
18. Eric M. Kramer. Computer models of auxin transport: a review and commentary. *J. Exp. Bot.*, 59(1):45–53, 2008.
19. Pawel Krupinski and Henrik Jnsson. Modeling auxin-regulated development. *Cold Spring Harb. Perspect. Biol.*, 2(2):a001560, 2010.
20. Cris Kuhlemeier. Phyllotaxis. *Trends in Plant Science*, 12(4):143–150, 2007.
21. Ottoline Leyser. The fall and rise of apical dominance. *Curr. Opin. Genet. Dev.*, 15(4):468–471, 2005.
22. Ottoline Leyser. The control of shoot branching: an example of plant information processing. *Plant Cell Environ.*, 32(6):694–703, 2009.
23. G. Mitchison. A model for vein formation in higher plants. *Pro. Roy. Soc. Lond. B.*, 207:79–109, 1980.
24. G. Mitchison. The polar transport of auxin and vein patterns in plants. *Phil. Trans. Roy. Soc. Lond.*, 295:461–471, 1981.
25. G. J. Mitchison. The effect of intracellular geometry on auxin transport II. Geotropism in shoots. *Proc. Roy. Soc. Lon. B.*, 214(1194):69–83, 1981.
26. J. Odell, H. Parunak, and B. Bauer. Extending UML for agents. In *Proc. AOIS Worshop at AAAI, Austin, 2000*, pages 3–17, 2000.
27. OMG. Maintainer of the UML standards. *http://www.omg.org*, 2008.
28. Veronica Ongaro and Ottoline Leyser. Hormonal control of shoot branching. *J. Exp. Bot.*, 59(1):67–74, 2008.
29. Tomasz Paciorek, Eva Zazímalová, Nadia Ruthardt, Jan Petrásek, York-Dieter Stierhof, Jürgen Kleine-Vehn, David A. Morris, Neil Emans, Gerd Jürgens, Niko Geldner, and Jiří Friml. Auxin inhibits endocytosis and promotes its own efflux from cells. *Nature*, 435(7046):1251–1256, 2005.
30. Geraint Parry, Alan Marchant, Sean May, Ranjan Swarup, Kamal Swarup, Nick James, Neil Graham, Trudie Allen, Tony Martucci, Antony Yemm, Richard Napier, Ken Manning, Graham King, and Malcolm Bennett. Quick on the uptake: Characterization of a family of plant auxin influx carriers. *Journal of Plant Growth Regulation*, 20(3):217–225, 2001.
31. Jan Petrášek, Jozef Mravec, Rodolphe Bouchard, Joshua J. Blakeslee, Melinda Abas, Daniela Seifertová, Justyna Wiśniewska, Zerihun Tadele, Martin Kubeš, Milada Čovanová, Pankaj Dhonukshe, Petr Skůpa, Eva Benková, Lucie Perry, Pavel

Křeček, Ok Ran Lee, Gerald R. Fink, Markus Geisler, Angus S. Murphy, Christian Luschnig, Eva Zažímalová, and Jiří Friml. PIN proteins perform a rate-limiting function in cellular auxin efflux. *Science*, 312(5775):914–918, 2006.

32. Przemyslaw Prusinkiewicz, Scott Crawford, Richard S. Smith, Karin Ljung, Tom Bennett, Veronica Ongaro, and Ottoline Leyser. Control of bud activation by an auxin transport switch. *Proc. Natl. Acad. Sci. USA*, 106(41):17431–36, 2009.
33. J. A. Raven. Transport of indoleacetic acid in plant cells in relation to pH and electrical potential gradients, and its significance for polar IAA transport. *New Phytologist*, 74(2):163–172, 1975.
34. Mark Read, Jon Timmis, Paul S. Andrews, and Vipin Kumar. A domain model of experimental autoimmune encephalomyelitis. In *CoSMoS 2009, York, UK, August 2009*, pages 9–44. Luniver Press, 2009.
35. Didier Reinhardt, Eva-Rachele Pesce, Pia Stieger, Therese Mandel, Kurt Baltensperger, Malcolm Bennett, Jan Traas, Jiří Friml, and Cris Kuhlemeier. Regulation of phyllotaxis by polar auxin transport. *Nature*, 426(6964):255–260, 2003.
36. P. H. Rubery and A. R. Sheldrake. Carrier-mediated auxin transport. *Planta*, 118(2):101–121, 1974.
37. T. Sachs. Polarity and the induction of organized vascular tissues. *Ann. Bot.*, 33(2):263–275, 1969.
38. T. Sachs. Patterned differentiation in plants. *Differentiation*, 11(1-3):65–73, 1978.
39. T. Sachs. The control of the patterned differentiation of vascular tissues. *Adv. Bot. Res. inc Adv. Plant Path.*, 9:151–262, 1981.
40. Michael Sauer, Jozef Balla, Christian Luschnig, Justyna Wiśniewska, Vilém Reinöhl, Jiří Friml, and Eva Benková. Canalization of auxin flow by Aux/IAA-ARF-dependent feedback regulation of PIN polarity. *Genes Dev.*, 20(20):2902–2911, 2006.
41. Enrico Scarpella, Danielle Marcos, Jiří Friml, and Thomas Berleth. Control of leaf vascular patterning by polar auxin transport. *Genes Dev.*, 20(8):1015–1027, 2006.
42. C. Uggla, T. Moritz, G. Sandberg, and B. Sundberg. Auxin as a positional signal in pattern formation in plants. *Proc. Natl. Acad. Sci. USA*, 93(17):9282–9286, 1996.
43. Anne Vieten, Michael Sauer, Philip B. Brewer, and Jiří Friml. Molecular and cellular aspects of auxin-transport-mediated development. *Trends Plant Sci.*, 12(4):160–168, 2007.
44. K. Webb and T. White. UML as a cell and biochemistry modeling language. *BioSystems*, 80:283–302, 2005.
45. Justyna Wiśniewska, Jian Xu, Daniela Seifertová, Philip B. Brewer, Kamil Růžička, Ikram Blilou, David Rouquié, Eva Benková, Ben Scheres, and Jiří Friml. Polar PIN localization directs auxin flow in plants. *Science*, 312(5775):883, 2006.