# One Modelling Formalism & Simulator Is Not Enough! A Perspective for Computational Biology Based on JAMES II

Adelinde M. Uhrmacher, Jan Himmelspach, Matthias Jeschke, Mathias John, Stefan Leye, Carsten Maus, Mathias Röhl, and Roland Ewald

University of Rostock
18059 Rostock, Germany
Albert-Einstein-Str. 21, D-18059 Rostock, Germany
lin@informatik.uni-rostock.de

**Abstract.** Diverse modelling formalisms are applied in Computational Biology. Some describe the biological system in a continuous manner, others focus on discrete-event systems, or on a combination of continuous and discrete descriptions. Similarly, there are many simulators that support different formalisms and execution types (e.g. sequential, parallel-distributed) of one and the same model. The latter is often done to increase efficiency, sometimes at the cost of accuracy and level of detail. JAMES II has been developed to support different modelling formalisms and different simulators and their combinations. It is based on a plug-in concept which enables developers to integrate spatial and non-spatial modelling formalisms (e.g. STOCHASTIC $\pi$ CALCULUS, BETA BINDERS, DEVS, SPACE-$\pi$), simulation algorithms (e.g. variants of Gillespie's algorithms (including Tau Leaping and NEXT SUBVOLUME METHOD), SPACE-$\pi$ simulator, parallel BETA BINDERS simulator) and supporting technologies (e.g. partitioning algorithms, data collection mechanisms, data structures, random number generators) into an existing framework. This eases method development and result evaluation in applied modelling and simulation as well as in modelling and simulation research.

## 1 Introduction

*A Model (M) for a system (S) and an experiment (E) is anything to which E can be applied in order to answer questions about S.* This definition that has been coined by Minsky in 1965 [Min65] implies the co-existence of several models for any system. Each model and its design is justified by its specific objectives. Simulation on the other hand can be interpreted as "an experiment performed at a model", as stated by Korn and Wait [KW78]. The term simulation is sometimes used for one simulation run, but more often it refers to the entire experimental setting including many simulation runs and the usage of additional methods for optimization, parameter estimation, sensitivity analysis etc. Each run requires a multitude of steps: e.g., model selection, initialization, defining the observers, selecting the simulation engine, and storing results – to name only a few. Given the long tradition of modelling and simulation, its many facets and the

J. Fisher (Ed.): FMSB 2008, LNBI 5054, pp. 123–138, 2008.

diversity of application areas, it is not surprising that a plethora of different modelling and simulation methods have been developed, and scarcely less simulation tools.

While modelling and simulation has been applied to gain a better understanding of biological systems for well over four decades, broad interest in applying modelling and simulation in cell biology has been renewed by recent developments in experimental methods, e.g. high content screening and microscopy, and has spawned the development of new modelling and simulation methods. Since the millennium, the significance of stochasticity in cellular information processing has become widely accepted, so that stochastic discrete-event simulation has emerged as an established method to complement conventional ordinary differential equations in biochemical simulations. This is also reflected in simulation tools for systems biology that have started to offer at least one discrete-event simulator in addition to numerical integration algorithms, e.g. [ROB05, TKHT04]. The trend to offer more flexibility is not exclusive to the simulation layer. Different parameter estimation methods [HS05] and different possibilities to describe models (e.g., by rules or with BETA BINDERS [GHP07]), are receiving more and more attention as well. Thus, the insight is taking hold in the computational biology realm that a silver bullet does not exist – there are only horses for courses. This diversification and the implied need for a flexible simulation framework is likely to increase over the next years, particularly as, in addition to noise, space is entering the stage of computational biology. In vivo experiments revealed that many intra-cellular effects depend on space, e.g. protein localization, cellular compartments, and molecular crowding [Kho06]. Approaches that support both stochasticity and space are therefore particularly promising [TNAT05, BR06].

The motivation for developing JAMES II (JAva-based Multipurpose Environment for Simulation) has been to support diverse application areas and to facilitate the development of new modelling and simulation methods. JAMES II has been created based on a "Plug'n simulate" [HU07b] concept which enables developers to integrate their ideas into an existing framework and thus eases the development and the evaluation of methods. In the following we will describe basic concepts and some of the current developments to support cell-biological applications in JAMES II.

## 2 JAMES II– Plug'n Simulate

The simulation framework JAMES II is a lean system consisting of a set of core classes. The core of JAMES II is the central and most rarely changed part of the framework. The main parts are: User interface, Data, Model, Simulator, Simulation, Experiment, and Registry. We used common software engineering techniques for the creation of the framework, e.g. the model-view-controller paradigm [GHJV95] for decoupling the parts, and the abstract factory and factory patterns [GHJV95] for realizing the "Plug'n simulate" approach. Another important design decision was to split model and simulation code completely. Thus, a simulator can access the interface of a model class but a model class is never allowed to access something in a simulator class. This makes it possible to switch the simulation engine (even during runtime) and to exchange the data structures used for the executable models – an essential feature for a flexible framework. In combination with an XML-based model component plugin, this flexibility enables

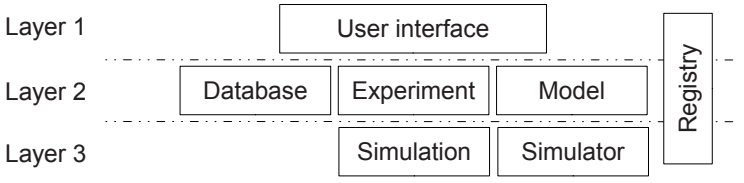


**Fig. 1.** Packages of the simulation framework JAMES II [HU07b]

the freedom of choice in regards to model data type, simulator code (algorithm as such or parts of the simulation algorithms, e.g. event queues), visualization, and runtime environment. The architecture is sketched in Figure 1. The layers depict the distance of a user from the packages.

Functionality not included in the core classes, especially modelling formalisms and simulation algorithms, can be extended by using plugins. Due to the strict separation between models and simulators, simulation algorithms can be easily exchanged and thus evaluated. This makes the PlugIn mechanism a base for a reliable evaluation of new simulation algorithms. For the integration of a new formalism, one has to create model classes which can represent an instantiated and executable model defined in a certain formalism. Conducting experiments requires at least one additional plugin that provides a simulator. Having created the formalism classes, one can directly start to code models and experiment with them. A prototypical example can be found in [HU07b], where cellular automata are added to the framework. Plugins exist for a number of formalisms, among them variants of DEVS [ZPK00], e.g. ML-DEVS [UEJ+07], and variants of the $\pi$ CALCULUS, e.g. BETA BINDERS [PQ05] and SPACE-$\pi$ [JEU08]. If models shall be described in a declarative manner, a model reader can be used, which converts arbitrary model definitions (e.g., from XML files or databases) into executable models [RU06] based on consistency checking of interface descriptions [RM07]. Interface definitions are according to the Unified Modelling Language 2.0 [OMG05]. Thereby, the provisions and requirements of each model component can be explicitly specified, internal details of a component, i.e. the implementation of model behaviour, can be hidden, and direct dependencies between models can be eliminated.

Different formalisms may require different simulation algorithms for their execution. Different hardware infrastructures (e.g. clusters, workstations, the Grid) may impose restrictions or options which should be taken into account by an algorithm. For example, symmetric multiprocessor machines provide fast access to shared memory. Even models described in the same formalism might require different simulators for an efficient simulation, depending on model size and other characteristics. Thus, various simulation plugins have been implemented for JAMES II. As they all provide simulators, i.e. algorithms that execute models, they are integrated via the `simulator` extension point. An extension point subsumes all plugins of one type and provides mechanisms to select the right plugin for a given problem (see fig. 2).

Other extension points provide partitioning and load balancing, random number generation and probability distributions, optimization, parameter estimation, data storage and retrieval, and experiment definitions. JAMES II also offers several extension points

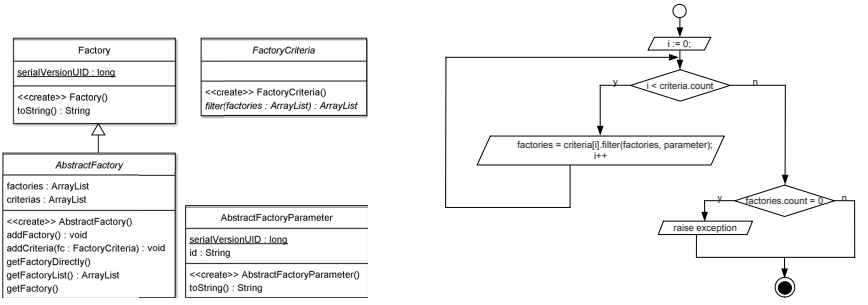


**Fig. 2.** Factory classes and flow-chart describing the plugin filtering process

for the integration of data structures, among them an extension point for the integration of event queues. Extension points can be defined easily, so their number and the number of implemented plugins grows steadily.

The `simulator` extension point can be combined with other extension points and forms the basis of a flexible and scalable experimentation layer. Such an experimentation layer can adapt to different types of experiments (e.g. optimization, validation), to different execution schemes (e.g. sequential, parallel distributed), and to different models realized in different modelling formalisms.

The selection process for a simulation algorithm is illustrated in figure 2. Several criteria are subsequently applied to select a suitable simulation algorithm. The first criterion is built-in and selects factories according to the user parametrization of the run. Each plugin is described by a list of factories and a name [HU07b]. For example, selecting a suitable simulator requires to apply a set of specific criteria for factories that can instantiate those for the model at hand. Such a factory also needs to suit the given partition (i.e. whether to compute the model in a distributed or sequential manner), and finally the most efficient factory shall be determined. For this, we strive to take earlier experiences regarding simulator performance into account [EHUar] (see section 4.2).

Experiments with JAMES II show that exchangeable algorithms are the precondition for an efficient execution of experiments [HU07a]. In addition, a subset of these algorithms are reusable across simulation algorithms, even if they have been designed for different formalisms. Thus, the implementation effort required for the creation of algorithms is significantly reduced, see e.g. [HU04]. In addition, the validation and evaluation of simulation algorithms is eased in such a framework: a novel algorithm simply has to be added to the framework and can be compared to all competing solutions, without bias or the need to recode those. The benefits of reusing simulation algorithms or data structures is similar to the arguments that pushes the development of model components. For example, if we define the nucleus as one model component, this allows us to easily replace the model by more abstract, refined, or alternative versions. Thus, a space of interesting hypotheses referring to the system under study can be evaluated without the need to recode the complete model. This freedom of choice with respect to model and simulator configuration is accompanied by an explicit representation of the experimental setting in XML, ensuring repeatability of simulation runs and comparability of the results achieved by alternative models.

## 3 Modelling Formalisms Supported in JAMES II

Modelling means to structure our knowledge about a given system. Thereby, the modelling formalism plays a crucial role. Whereas continuous modelling formalisms support a macro, population-based view, with its associated continuous, deterministic dynamics progressing at the same speed and being based on real-valued variables, discrete event approaches in contrast tend to focus on a micro perception of systems and their discrete, stochastic dynamics. Variables can be arbitrarily scaled. Sub-systems advance over a continuous time scale, but in steps of variable size. Discrete event models forego assumptions about homogeneous structure or behaviour. For example, DEVS [ZPK00], PETRI NETS [Mur89], STATE CHARTS [Har87] and STOCHASTIC $\pi$ CALCULUS [Pri95] are formal and generally applicable approaches toward discrete event systems modelling. Their use has been explored in Systems Biology (e.g. [PRSS01, FPH+05, EMRU07]), and depending on their success inspired a broader exploitation and the refinement of methods. The focus in JAMES II has been on discrete event formalisms, with emphasis on DEVS, followed by STOCHASTIC $\pi$ CALCULUS; lately a first plugin for STATE CHARTS has been implemented.

### 3.1 Biologically Inspired Variants of DEVS

The strength of DEVS lies in its modular, hierarchical design which provides a basis for developing complex cellular models, whose individual entities are easy to understand if visualized, e.g. by a STATE CHARTS variant. In contrast to STATE CHARTS, the interaction between individual components is explicitly modelled and the interfaces between systems are clearly described by input and output ports. DEVS supports a parallel composition of models via coupling. The general structure of DEVS models is static, a problem that it shares with STATE CHARTS and PETRI NETS. The generation of new reactions, new interactions, or new components is therefore not supported by default. Starting already in the 1980s, several extensions have addressed the problem of variable structures, among them recent developments like DYNDEVS, $\rho$-DEVS, and ML-DEVS, which are realized as plugins in JAMES II.

DYNDEVS [Uhr01] is a reflective variant of DEVS which supports dynamic behaviour, composition, and interaction patterns. In $\rho$-DEVS [UHRE06] dynamic ports and multi-couplings are introduced, whose combination allows models to reflect significant state changes to the outside world and to enable or disable certain interactions at the same time. Unlike DEVS, which realizes an extensional definition of couplings between individual ports, multi-couplings in conjunction with variable ports form an elegant mechanism of dynamic coupling. ML-DEVS [UEJ+07] is our most recent addition to the family of DEVS-formalisms. It inherits from $\rho$-DEVS variable structures, dynamic ports, and multi-couplings. In addition, it supports an explicit description of macro and micro level. Therefore, the coupled model is equipped with a state and a behaviour of its own, such that the macro level does not appear as a separate unit (an executive) of the coupled model. Secondly, information at macro level can be accessed from micro level and vice versa, which is realized by value couplings and variable ports. Downward and upward activation can be modelled by synchronous activation of multiple micro models and invariants at macro level. Current work is dedicated to applying the formalism to RNA secondary structure prediction and associated regulatory mechanisms.

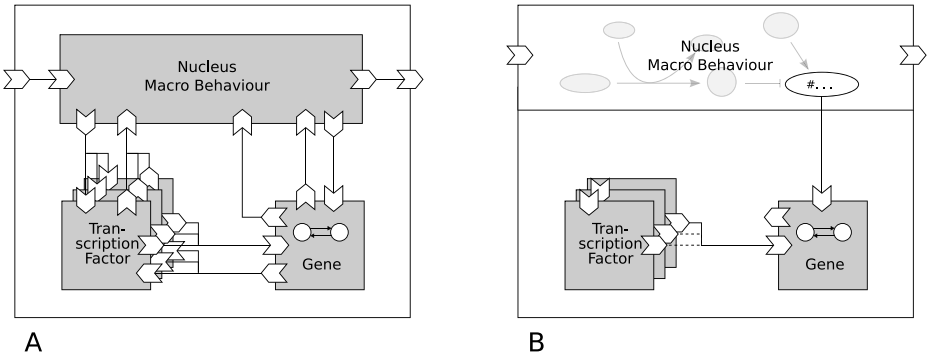


**Fig. 3.** Comparison of multi-level modelling with DEVS and ML-DEVS. (A) With DEVS the macro level behaviour is modelled at the same hierarchical level as the micro level models. (B) With ML-DEVS the macro dynamics are part of the coupled model. Functions for downward and upward causation reduce the number of explicit couplings needed.

Whereas the biologically inspired extensions of the DEVS formalisms aim to soften the original rigid, modular and hierarchical structure of the formalism, e.g. by introducing variable structures, variable ports, and multi-level interactions, the opposite development can be observed for other formalisms which introduce additional means to structure biological models. These structures are often spatially interpreted, e.g. in terms of cell membranes or compartments. For example, this is the case for BETA BINDERS, which base on STOCHASTIC $\pi$ CALCULUS.

### 3.2 Taking Space into Account: SPACE-$\pi$

BETA BINDERS [PQ05], as well as MEMBRANE-CALCULI [Car03] or BIOAMBIENTS [RPS+04], address the need to structure space into compartments and confine processes and their interactions spatially. Here, the focus is on indirect, relative space. Given the experimental setups, e.g. confocal microscopy, biologists are particularly interested in a combination of the individual-based approach with absolute space. This has been the motivation for developing SPACE-$\pi$ [JEU08]. SPACE-$\pi$ extends the $\pi$ CALCULUS in a way such that processes are associated with coordinates and individual movement functions. The coordinates are related to a given vector space that provides a norm for calculating distances. A minimum distance is assigned to each channel. As usual, processes can communicate over a channel, but as an additional requirement their distance must be less than or equal to the minimum distance of the channel. Processes move continuously between interactions, as determined by their movement functions. Therefore, SPACE-$\pi$ is considered to be a hybrid modelling formalism.

Introducing space to the $\pi$ CALCULUS seems to be straightforward, as it just requires some further restrictions to the standard communication rule. However, an explicit notion of time is needed to describe movement functions. Therefore, an essential component of SPACE-$\pi$ is a timed version of the $\pi$ CALCULUS. Although many timed process algebras already exist, the common concept of extending process algebras with time as presented in [NS92] cannot be used. This is because on one hand SPACE-$\pi$ requires communication to occur as soon as possible, e.g. to describe colliding molecules. On

the other hand, the time of an interaction strongly depends on process motion. Since processes can change their motion by communication, one interaction may have an impact on all future events. Therefore, the essential assumption of timed process algebras to consider processes one by one and let them idle for a certain amount of time does not hold in case of SPACE-$\pi$. Instead, to assure a valid execution, the global minimum time for the next reaction needs to be calculated after every communication. This makes the approach rather unique in the context of timed process algebras.

The advantage of the SPACE-$\pi$ approach is that intracellular structures and spatial effects can be represented in a very detailed manner. It is possible to build the trails of membranes or microtubules, to introduce compartments and active transportation processes. Even the impact of molecule sizes and shapes can be modelled. This makes the approach applicable to scenarios that are hard to model with implicit representations of space.

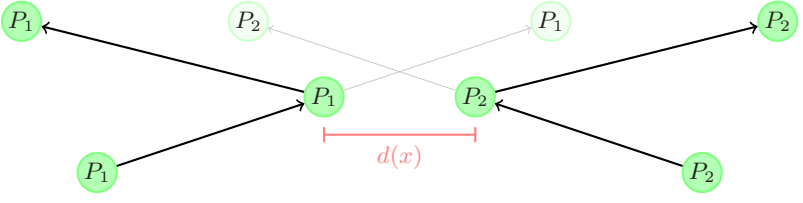


**Fig. 4.** Communication of two processes: $P_1$ and $P_2$ move along some vectors given by their movement functions. If one process is sending and the other receiving on channel $x$, they can communicate because at some point in time their distance is less than the channel distance $d(x)$. Thereby, they change their movement functions such that they move along different vectors after communicating. Although depicted as simple vectors here, SPACE-$\pi$'s movement functions can also be used to describe more complex continuous motions.

### 3.3 Summary

Several plugins exist for various formalisms, including ML-DEVS, SPACE-$\pi$, STOCHASTIC $\pi$ CALCULUS, BETA BINDERS, and chemical reaction networks (described by reaction rules). Although the supported formalisms are quite different, their realization has been facilitated by the core classes in JAMES II and also a reuse of data structures across different formalisms has reduced the implementation effort (see section 5 for further details). However, the architecture and the "Plug'n simulate" concept of JAMES II are even more significant for developing simulation algorithms. For each modelling formalism there is at least one simulator plugin, but many are supported by several simulation algorithms. Up to now, no "multi formalism" approaches are integrated into JAMES II, although such an integration would not pose any problems for the architecture. Some work in this direction is under way, e.g. to combine $\pi$ CALCULUS and DEVS [MJU07]. These approaches will be strictly separated from the existing realizations (besides the aforementioned reuse of sub-algorithms and data structures) to ensure an efficient execution.

## 4 Simulation Engines for Computational Biology in JAMES II

The simulation layer of JAMES II focuses on the discrete event world, although some plugins for numerical integration and hybrid simulation exist as well. Gillespie's algorithms [Gil77] and their variants are of particular interest for the area of systems biology and their implementation in JAMES II reflects the idea of re-usable simulation algorithms and data structures. In the following, these and a similar set of process algebra simulators will serve as examples.

In [Gil77], Gillespie introduced two methods to stochastically simulate reaction networks, the Direct Method and the First Reaction Method. Although both methods produce the same results, their performance may differ strongly. This is caused by the use of different operations. For example, the Direct Method needs to generate only two random numbers per iteration, whereas the First Reaction variant needs $r + 1$ random numbers per iteration, $r$ being the number of reactions. Gibson and Bruck propose some enhancements to Gillespie's original algorithms in [GB00], the Next Reaction Method. The new method reduces the time-consuming re-calculation of reaction propensities. A dependency graph is used to identify the reactions that require a propensity update. Furthermore, Gibson and Bruck's approach linearly interpolates the reaction times of all updated reactions, so that the generation of additional random numbers is avoided. Although all SSA approaches work well for small systems and deliver exact stochastic results, they do not perform well when applied to larger problems. This is especially true for systems with concurrent reactions of differing speed (e.g., gene expression and metabolic reactions): If populations of the metabolites are sufficiently high, many iterations (and propensity updates) are needed without any significant changes in any reaction propensity. This problem is equivalent to the challenge of numerically integrating stiff ODE systems. To overcome this problem in the discrete-event realm, a technique called Tau Leaping has been introduced by Gillespie et al. [Gil01]. Tau Leaping approximates the execution of Gillespie's exact approach by leaping forward a time step $\tau$, in which the propensities of all reactions are *approximately* constant. How often each reaction has occurred during this leap can be determined by a Poisson distribution. All reaction occurrences are then executed at once, their propensities are updated, and the algorithm continues by determining the size of the $\tau$ leap for the next iteration. One has to solve several problems to implement a suitable Tau Leaping method, as outlined in [TB04, CGP06]. The Direct Method, two implementation variants of the Next Reaction Method, and Tau Leaping have been realized as plugins in JAMES II. They operate on the same model interface and re-use auxiliary data structures like event queues, etc.

### 4.1 Taking Space into Account

Different approaches toward spatial simulation exist [TNAT05]. At the microscopic level, the fields of molecular dynamics [vGB90] and Brownian dynamics permit the accurate simulation of single interacting particles in continuous space. However, their high computational effort hampers their applications to large scale models containing many thousand particles. Rather than considering individual particles and instead focusing on concentrations of species, partial differential equations operate on a macroscopic level with continuous space and time. As the approach is deterministic, stochastic effects cannot be taken into account easily. The basic algorithm for simulating reactions

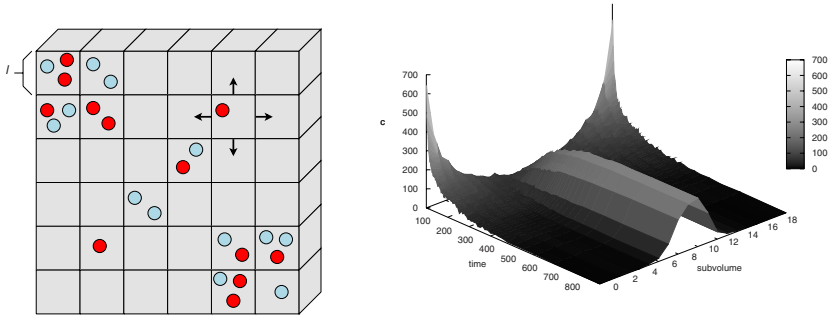


**Fig. 5. Left**: Discretization of a volume into smaller sub-volumes with side length $l$. The diffusion of molecules between neighbouring sub-volumes is modelled as a Markov process with $d_i \Delta t = \frac{D_i}{l^2} \cdot \Delta t$ as the probability that a particular molecule of species $i$ performs a diffusion during the infinitesimal small time step $\Delta t$ (with $D_i$ as the diffusion constant for species $i$). **Right**: The concentration for sub-volumes of a simple model is plotted for different time stamps. The model consists of a row of 20 sub-volumes with an initial distribution of 1000 molecules of species *A* in sub-volume 0 and 1000 molecules of species *B* in sub-volume 19 and a consuming reaction $A+B \rightarrow C$. Molecules *A* and *B* diffuse towards the centre where they react and the concentration of molecule *C* increases.

between chemical species on a mesoscopic level (no individuals, but discrete amount of species elements) is the already mentioned stochastic simulation algorithm (SSA) by Gillespie [Gil77]. One key assumption is that the distribution of the species inside the volume is homogeneous. To simulate systems that do not adhere to this assumption, other approaches that allow to consider compartments and the diffusion of species are necessary, e.g. [Kho06].

A common way of introducing diffusion on mesoscopic level is the partition of space into sub-volumes and the extension of the master equation with a diffusion term, resulting in the reaction-diffusion master equation (RDME) [Gar96]. The solution of the RDME is intractable for all but very simple systems, leading to the development of the Next-Subvolume Method (NSM) [EE04], an algorithm that generates trajectories of an underlying RDME, similarly to SSA sampling the chemical master equation. The Next-Subvolume Method is a discrete-event algorithm for simulating both reactions and diffusion of species within a volume in which particles are distributed inhomogeneously. The volume is partitioned into cubical sub-volumes, each representing a well-stirred system. Events generated by the algorithm are either reactions inside or diffusions between these sub-volumes. A plugin for NSM has been realized in JAMES II. NSM uses a variant of Gillespie's Direct Method to calculate the first event times for all sub-volumes during initialization. Within the main loop, the sub-volume assigned to the smallest next event time is selected and the current event type according to the diffusion and reaction rates is determined. Finally, the event is executed and an update of the model state occurs. Note that the state update is performed only in a small region of the model volume, because either one sub-volume (in the case of a reaction) or two sub-volumes (in the case of a diffusion) are affected and their propensities and next event times have to be updated. With its discretization of space into sub-volumes, the

Next-Subvolume Method lends itself to a distributed parallel execution by assigning sets of sub-volumes to logical processes. Messages exchanged between logical processes correspond to diffusion events involving neighbouring sub-volumes that reside on different logical processes. Current work is directed towards exploring the specific requirements and potential of an optimistic NSM version on a grid-inspired platform and first results of our experiments with an optimistic NSM variant are reported in [JEP+ar]. Our developments based on NSM have not been driven by specific modelling formalisms – the modelling formalisms for the Gillespie variants are simple reaction networks, which have been enriched for the NSM simulator by assigning diffusion coefficients to the species and information about their spatial distribution. The focus has been on the simulation layer taking Gillespie's approach as a starting point. The development of other simulators has been motivated by supporting specific modelling formalisms.

As STOCHASTIC $\pi$ CALCULUS, SPACE-$\pi$, and BETA BINDERS have a common root, basic plugins to support the $\pi$ CALCULUS have been added to JAMES II. They form a common base for all simulators processing variants of the $\pi$ CALCULUS. The SPACE-$\pi$ formalism locates and moves individual processes in absolute space, hence its simulator works in an individual-based manner. With movement functions that are defined to be continuous and could, in principle, even be defined by differential equations, the formalism is clearly hybrid. The hybrid state comprises, besides the processes themselves, all processor positions which change continuously over time. Obviously, this situation provides plenty of challenges for an efficient execution of larger SPACE-$\pi$ systems. Hence, we developed and implemented an algorithm that approximates movement functions by a sequence of vectors, each of them interpolating the function for $\delta_t$ time, resulting in a discretization of the continuous movement of the processes into a sequence of uniform motions. As $\delta_t$ can be chosen arbitrarily, the approximation's fidelity can be adjusted seamlessly [JEU08]. The advantage of the simulation is its decent computational complexity, however, if more complex movement functions are involved or a very precise result is necessary, the current solution will not be sufficient and other algorithms will be needed, e.g. approximation methods that adapt their interval size to the slope of the movement functions at hand.

The modelling formalism BETA BINDERS (developed by [PQ05]) provides, similarly to DEVS, means to structure space relatively, e.g. to describe compartments or to distinguish between inter and intra-cellular behaviour by introducing boxes (so called BIOPROCESSES) and explicit interaction sites. A sequential simulator [HLP+06] and a parallel one have been developed for the BETA BINDERS formalism as JAMES II plugins [LPU07]. The stochastic interaction between BIOPROCESSES does not allow an easy definition of lookaheads, which moves the attention to optimistic parallel approaches. However, due to the large effort required in handling the model structure an unbounded optimistic simulation does not appear very promising either. Therefore, we decided to combine conservative and optimistic features in a parallel simulator. Whereas the intra events (that are taking place within one box) are processed in an optimistic manner, the inter events form a kind of barrier and as such are only processed if they are safe. The later is guaranteed by letting all BIOPROCESSES advance up to this barrier [LPU07]. As with all parallel, distributed approaches the question of how to best partition such a model arises, and is subject of ongoing work. JAMES II allows

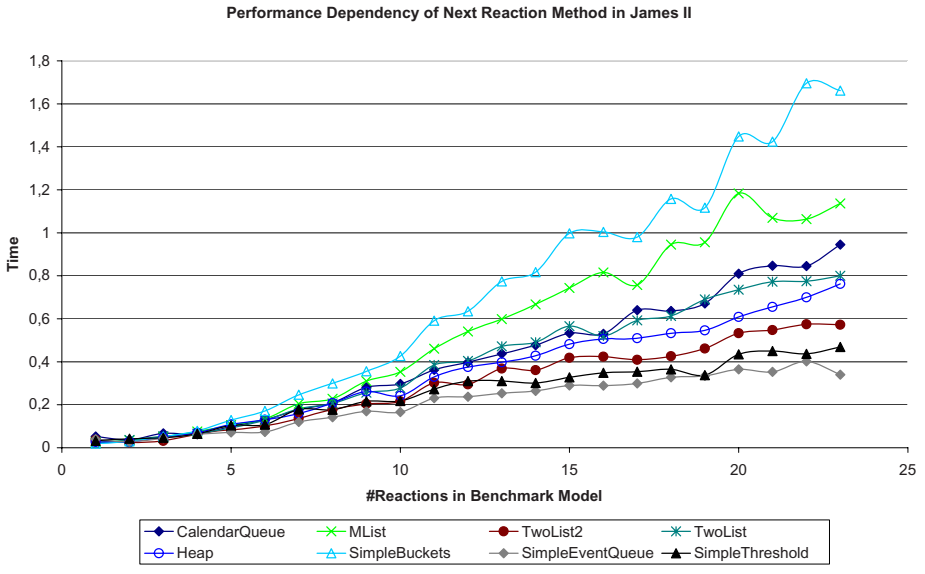


**Fig. 6.** Performance dependency of the Next Reaction Method regarding the event queue implementation (see [HU07a] for other examples and a discussion of the algorithms). Each setup was replicated 3 - 15 times.

to develop new simulators with very little effort. This is particularly true for simulator families, e.g. simulators for variants of modelling formalisms, e.g. DEVS [HU04] or $\pi$ CALCULUS. However, also simulators that are quite different share a lot of details. Their re-use reduces programming effort, errors, and, at the same time, increases the chance of an unbiased evaluation. As these details have typically also a significant impact on the efficiency of the simulation engine, they deserve a closer look.

### 4.2 Details Matter

Simulators are usually not built as monolithic algorithms, but consist of several sub-structures and sub-algorithms. These may have a strong impact on simulation speed and even the accuracy of the produced results (e.g. when considering approximation algorithms). Simulation algorithms may rely on event queues [HU07a], random number generators, or data structures to manage the simulator's state. Parallel and distributed simulators may also require partitioning or load balancing algorithms.

JAMES II provides several plugins for most of those sub-algorithms, including random number generators, probability distributions, event queues, and partitioning algorithms. As an illustration, consider the average runtime of the Next Reaction Method implementation of JAMES II (see section 4) when using different event queue implementations (figure 6).

The benchmark model was a simple 10-species reaction network with arbitrarily many reactions of form $X_i + X_{i+1} \rightarrow X_{i+2}$, but the performance difference is still

considerable ($\approx 400\%$ when choosing SimpleEventQueue instead of SimpleBuckets) – even considering different implementations of the *same* algorithm (cf. TwoList and TwoList2 in fig. 6).

Thus, data structures like event queues are of essential importance when evaluating discrete-event simulators. They influence the performance of the simulation engine and may in turn be influenced by certain properties of the model. For example, the performance of an event queue may depend on the distribution of event time stamps. These interdependencies are a general problem in assessing algorithm performance. The design of JAMES II facilitates algorithm performance evaluation and development of new simulation methods by allowing for isolated testing within a fixed, unbiased experimentation environment. The "Plug'n simulate" concept complements this idea by providing means to add new functionality to the system [HU07b].

Partitioning is another domain where sub-algorithms play an important role. Model partitioning, i.e. distributing model entities over the set of available processors, is a prerequisite for any distributed simulation. It aims at minimizing communication between parallel simulation processes and assigning each processor a fair share of entities. Doing it badly may hamper the performance of a distributed simulation algorithm to the point of slower-than-sequential simulation speed. JAMES II provides a partitioning layer [EHU06] to experiment with different partitioning strategies. Several partitioning algorithms have been integrated as plugins (including a wrapper for the METIS [KK98] package) and have been evaluated. Again, experiments showed that the performance gain from using a particular partitioning strategy depends on the simulation algorithm, the model, and the available infrastructure.

Simulation experiments in computational biology are typically not restricted to single simulation runs. They may require thousands of replications and could even include parameter estimation or optimization methods on top (for which again a multitude of methods exist), so that performance issues become even more pressing. Additionally, new algorithms are frequently proposed and need to be evaluated. For example, Cao et al. introduced an optimized version of Gillespie's Direct Method, which is faster than the Next Reaction Method on many problem instances [CLP04]. JAMES II allows to revalidate these findings with custom event queue implementations, since these may have a huge impact on the overall performance [CLP04]. Simulation algorithms that have been built upon the SSA approach, such as the Next Subvolume Method (see section 4), will also benefit from such re-validations, because performance analysis could direct implementation efforts to the most promising code optimizations. These challenges motivated the creation of an infrastructure for simulation algorithm performance analysis in JAMES II [EHUar], which should lead to effective algorithm selection mechanisms in the future.

## 5 Benefits of JAMES II

The integration of so many aspects into a single modelling and simulation framework requires considerable additional software engineering efforts. From our experiences, these efforts are vastly compensated by time savings from reuse and other benefits, such as unbiased algorithm comparison and validation.

JAMES II provides reuse on different levels. If an additional modelling paradigm shall be supported, only the modelling formalism and at least one simulation algorithm have to be added. Plugins from other extension points, e.g. the experimentation layer with algorithms for optimization, random number generation, partitioning, data sinks and so on, can be reused without any further effort. The actual development of new plugins is supported by predefined algorithms and data structures. For example, simulation algorithms that rely on event queues simply reuse the validated and evaluated ones from the corresponding JAMES II extension point. This reduces the effort for developing new modelling formalisms and simulators significantly.

The integration of new simulation algorithms for existing formalisms is eased as well. Often only small variations between simulators that execute a formalism – e.g. sequentially, in parallel, paced, or unpaced – do exist. To generate a paced variant from an unpaced one usually requires merely a few lines of code, e.g. the Template pattern helped us to develop a set of simulators for DEVS with very little effort [HU04]. Newly developed algorithms can be evaluated and validated easily by comparing their results with those of existing algorithms. Does the new algorithm produce the same results? For which type of model and available infrastructure works the new algorithm best? Developing simulators for new variants of already supported modelling formalisms, e.g. ML-DEVS or SPACE-$\pi$, also requires less effort, because parts of existing simulators can still be reused. This is also helpful to identify essential parts and re-occurring patterns in simulation algorithms, which facilitates the conception of new simulation engines.

New data structures and algorithms become available in all situations where their corresponding extension points are used. If a new random number generator or a new event queue is available, it can be directly used in all relevant settings, e.g. when experimenting with stochastic and discrete event models. Thus, new developments are instantly propagated to all potential users. The development of particular data structures and algorithms can be done by domain experts, who have the required knowledge and experience. This is very important, as the field of modelling and simulation subsumes a variety of different expertises. These range from in-depth knowledge about mathematics (random number generation, sensitivity analysis, optimization, statistics, graph theory) and computer science (databases, compilers, efficient data structures and algorithms in general) up to application-specific extensions required for certain domains, e.g. the use of metaphors in biology.

## 6 Conclusion

The purpose of JAMES II is twofold. On the one hand its flexibility and scalability is aimed at supporting a wide range of different application areas and different requirements. This feature has shown to be of particular value in research areas like Computational Biology, which is characterized by the wish to experiment with diverse model and simulation types. On the other hand JAMES II's virtue lies in facilitating the development and evaluation of new modelling and simulation methods.

Thus, to support application studies and methodological studies equally is the strength of JAMES II, which is also reflected in current projects. Some of them are directed towards applications, e.g. models of the Wnt signalling pathway, or gene regulatory

mechanisms. Others are concerned with the development of new modelling and simulation methods, including modelling formalisms like ML-DEVS and SPACE-$\pi$, or new simulators like the parallel optimistic versions for BETA BINDERS and NEXT SUBVOLUME METHOD, and the combination of modelling formalisms and simulator engines. Both endeavours are tightly connected, as new requirements drive the development of new methods. Both types of our experiments are directed towards a better understanding of complex systems, i.e. biological systems and simulation systems, and their many interdependencies.

## Acknowledgements

This research is supported by the DFG (German Research Foundation).

## References

[BR06] Broderick, G., Rubin, E.: The realistic modeling of biological systems: A workshop synopsis. ComPlexUs Modeling in Systems Biology, Social Cognitive and Information Science 3(4), 217–230 (2006)

[Car03] Cardelli, L.: Membrane interactions. In: BioConcur 2003, Workshop on Concurrent Models in Molecular Biology (2003)

[CGP06] Cao, Y., Gillespie, D.T., Petzold, L.R.: Efficient step size selection for the tau-leaping simulation method. J. Chem. Phys. 124, 044109 (2006)

[CLP04] Cao, Y., Li, H., Petzold, L.: Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. The Journal of Chemical Physics 121(9), 4059–4067 (2004)

[EE04] Elf, J., Ehrenberg, M.: Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. Syst. Biol (Stevenage) 1(2), 230–236 (2004)

[EHU06] Ewald, R., Himmelspach, J., Uhrmacher, A.M.: Embedding a non-fragmenting partitioning algorithm for hierarchical models into the partitioning layer of James II. In: WSC 2006: Proceedings of the 38th conference on Winter simulation (2006)

[EHUar] Ewald, R., Himmelspach, J., Uhrmacher, A.M.: An algorithm selection approach for simulation systems. In: Proceedings of the 22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008) (to appear, 2008)

[EMRU07] Ewald, R., Maus, C., Rolfs, A., Uhrmacher, A.M.: Discrete event modelling and simulation in systems biology. Journal of Simulation 1(2), 81–96 (2007)

[FPH+05] Fisher, J., Piterman, N., Hubbard, J., Stern, M., Harel, D.: Computational insights into C. elegans vulval development. PNAS 102(5), 1951–1956 (2005)

[Gar96] Gardiner, C.W.: Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences (Springer Series in Synergetics). Springer, Heidelberg (1996)

[GB00] Gibson, M.A., Bruck, J.: Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. J. Chem. Physics 104, 1876–1889 (2000)

[GHJV95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: elements of reusable object-oriented software. Addison-Wesley, Reading (1995)

[GHP07] Guerriero, M.L., Heath, J.K., Priami, C.: An automated translation from a narrative language for biological modelling into process algebra. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 136–151. Springer, Heidelberg (2007)

[Gil77] Gillespie, D.T.: Exact Stochastic Simulation of Coupled Chemical Reactions. The Journal of Physical Chemistry B 81(25), 2340–2361 (1977)

[Gil01] Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. The Journal of Chemical Physics (2001)

[Har87] Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8(3), 231–274 (1987)

[HLP+06] Himmelspach, J., Lecca, P., Prandi, D., Priami, C., Quaglia, P., Uhrmacher, A.M.: Developing an hierarchical simulator for beta-binders. In: 20th Workshop on Principles of Advanced and Distributed Simulation (PADS 2006), pp. 92–102. IEEE Computer Society, Los Alamitos (2006)

[HS05] Jirstrand, M., Schmidt, H.: Systems biology toolbox for matlab: A computational platform for research in systems biology. Bioinformatics (2005)

[HU04] Himmelspach, J., Uhrmacher, A.M.: A component-based simulation layer for james. In: ACM Press (ed.): PADS 2004: Proceedings of the eighteenth workshop on Parallel and distributed simulation, pp. 115–122. IEEE Computer Society, Los Alamitos (2004)

[HU07a] Himmelspach, J., Uhrmacher, A.M.: The event queue problem and pdevs. In: Proceedings of the SpringSim 2007, DEVS Integrative M&S Symposium, pp. 257–264. SCS (2007)

[HU07b] Himmelspach, J., Uhrmacher, A.M.: Plug'n simulate. In: Proceedings of the Spring Simulation Multiconference, pp. 137–143. IEEE Computer Society, Los Alamitos (2007)

[JEP+ar] Jeschke, M., Ewald, R., Park, A., Fujimoto, R., Uhrmacher, A.M.: Parallel and distributed spatial simulation of chemical reactions. In: Proceedings of the 22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008) (to appear, 2008)

[JEU08] John, M., Ewald, R., Uhrmacher, A.M.: A spatial extension to the pi calculus. In: Proc. of the 1st Workshop From Biology To Concurrency and back (FBTC 2007). Electronic Notes in Theoretical Computer Science, vol. 194, pp. 133–148 (2008)

[Kho06] Kholodenko, B.N.: Cell-signalling dynamics in time and space. Nature Reviews Molecular Cell Biology 7(3), 165–176 (2006)

[KK98] Karypis, G., Kumar, V.: MeTis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices (Version 4.0) (September 1998)

[KW78] Korn, G.A., Wait, J.V.: Digital continuous-system simulation. Prentice-Hall, Englewood Cliffs (1978)

[LPU07] Leye, S., Priami, C., Uhrmacher, A.M.: A parallel beta-binders simulator. Technical Report 17/2007, The Microsoft Research - University of Trento Centre for Computational and Systems Biology (2007)

[Min65] Minsky, M.: Models, minds, machines. In: Proc. IFIP Congress, pp. 45–49 (1965)

[MJU07] Maus, C., John, M., Uhrmacher, A.M.: A multi-level and multi-formalism approach for model composition in systems biology. In: Conference on Computational Methods in Systems Biology, Edinburgh, Poster (2007)

[Mur89] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–574 (1989)

[NS92] Nicollin, X., Sifakis, J.: An Overview and Synthesis on Timed Process Algebras. In: Larsen, K.G., Skou, A. (eds.) CAV 1991. LNCS, vol. 575, pp. 376–398. Springer, Heidelberg (1992)

[OMG05] OMG. UML superstructure specification version 2.0 (document formal/05-07-04) (July 2005), http://www.omg.org/cgi-bin/doc?formal/05-07-04

[PQ05] Priami, C., Quaglia, P.: Beta binders for biological interactions. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 20–33 Springer, Heidelberg (2005)

[Pri95] Priami, C.: Stochastic $\pi$-calculus. The Computer Journal 38(6), 578–589 (1995)

[PRSS01] Priami, C., Regev, A., Shapiro, E., Silvermann, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters 80, 25–31 (2001)

[RM07] Röhl, M., Morgenstern, S.: Composing simulation models using interface definitions based on web service descriptions. In: WSC 2007, pp. 815–822 (2007)

[ROB05] Ramsey, S., Orrell, D., Bolouri, H.: Dizzy: Stochastic simulation of large scale genetic regulatory networks. Journal of Bioinformatics and Computational Biology 01(13), 415–436 (2005)

[RPS+04] Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: BioAmbients: an abstraction for biological compartments. Theor. Comput. Sci. 325(1), 141–167 (2004)

[RU06] Röhl, M., Uhrmacher, A.M.: Composing simulations from xml-specified model components. In: Proceedings of the Winter Simulation Conference 2006, pp. 1083–1090. ACM, New York (2006)

[TB04] Tian, T., Burrage, K.: Binomial leap methods for simulating stochastic chemical kinetics. The Journal of Chemical Physics 121(10356), 10356–10364 (2004)

[TKHT04] Takahashi, K., Kaizu, K., Hu, B., Tomita, M.: A multi-algorithm, multi-timescale method for cell simulation. Bioinformatics 20, 538–546 (2004)

[TNAT05] Takahashi, K., Nanda, S., Arjunan, V., Tomita, M.: Space in systems biology of signaling pathways: towards intracellular molecular crowding in silico. FEBS letters 579(8), 1783–1788 (2005)

[UEJ+07] Uhrmacher, A.M., Ewald, R., John, M., Maus, C., Jeschke, M., Biermann, S.: Combining micro and macro-modeling in devs for computational biology. In: Proc. of the 2007 Winter Simulation Conference, pp. 871–880 (2007)

[Uhr01] Uhrmacher, A.M.: Dynamic structures in modeling and simulation - a reflective approach. ACM Transactions on Modeling and Simulation 11(2), 206–232 (2001)

[UHRE06] Uhrmacher, A.M., Himmelspach, J., Röhl, M., Ewald, R.: Introducing variable ports and multi-couplings for cell biological modeling in devs. In: Proc. of the 2006 Winter Simulation Conference, pp. 832–840 (2006)

[vGB90] van Gunsteren, W.F., Berendsen, H.J.: Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. Angewandte Chemie International Edition in English 29(9), 992–1023 (1990)

[ZPK00] Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation. Academic Press, London (2000)