Learning by Doing: Software Projects in CSE Education

Martin Bernreuther and Hans-Joachim Bungartz

[1] IPVS, Universität Stuttgart, Universitätsstr. 38, D-70569 Stuttgart, Germany
[2] Institut für Informatik, TU München, Boltzmannstr. 3, D-85748 Garching, Germany
Martin.Bernreuther@ipvs.uni-stuttgart.de

Abstract. Software development is one of the main routine activities in Computational Science and Engineering (CSE). Nevertheless, there is a huge gap between software engineering techniques available and established today in most fields where mainstream software is developed on the one hand and the typical extent of their application in a CSE context on the other hand. CSE curricula often reflect this tendency by not including software engineering topics adequately. This contribution reports experiences with a new course format called "student project" in the CSE master's program at TU München. There, for about half a year, a group of 4-8 students cooperate on a software development project – this time dealing with molecular dynamics. Although it is one objective to get a well performing code, the project's focus is on the consequent application of software engineering and project management practices.

Keywords: computational science and engineering, software engineering, education, molecular dynamics.

1 Introduction

If we look at both CSE curricula and CSE practice, there is, frequently, an obvious lack of software-related issues. This seems to be somewhat strange, esp. against the background that a significant part of a CSE researcher's or practitioner's everyday work is about making software (or programs, at least). The reasons of this gap are diverse: On the one hand, today, the focus of software engineering is a completely non-numerical one, while on the other hand most CSE people are strongly convinced that software engineering methods might be helpful elsewhere, but are definitely harmful to code performance. Recently, finally, the opinion that this continuing alienation might lead to a real simulation software crisis has got more and more support from the CSE side, too. First, experience and statistical investigations say that, typically, more than two thirds of software overall lifecycle cost is due to maintenance, only. Second, in the memorandum [1], classical software issues such as valdiation, verification, or quality management are identified as key issues for large-scale software in a CSE context, too. And third, the 2005 PITAC report on Computational Science [2]

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 161–168, 2006.

even speaks of a "crisis in computational science software" we are going to face or already facing, resp.

As a consequence of this development, CSE curricula must react and invest more in teaching selected aspects of software engineering, at least. Hence, at Technische Universität München (TUM), we decided to do a first step and to integrate a so-called student project into TUM's international CSE master's program [3]. The student project is a format adopted from Universität Stuttgart, where such projects have been a very successful part of the Software Engineering curriculum for several years [4] and have, recently, also been offered by the authors with a CSE-related topic (*Computational Steering - The Virtual Wind Tunnel* [5]). Of course, normally, planning to add something new to an existing study program is a quite complicated endeavour, since hardly anyone will be eager to give up some course as a countermove. However, we were lucky to have the necessary freedom, since our new honours program *Bavarian Graduate School of Computational Engineering (BGCE)* [6], a special offer to the best students of the CSE program and of two other master's programs, has a ten credit slot for some project work that had to be filled with life anyway, and we used exactly that slot for the new student project.

What are the essentials of this format? First of all, such a student project shall reflect the real-life process of making software as much as possible in an academic environment. For that, a team of, here, 4-8 students of, typically, different background gets the task of designing and developing a precisely specified program system in a limited amount of time (actually 6-8 months). Furthermore, not just the implementation, but several steps of the software development cycle have to be covered – from the very first concepts and design considerations up to the final handing-over of the well-documented code to the customer. This implies that there are different parts to be filled: the advisor, the examiner, and a customer on the instructors' side, as well as a project manager, a configuration manager, and several specialists such as a parallel programmer etc. on the students' side. Here, the explicit existence of a customer as a person not that much involved in the project, but nevertheless deciding about the amount of success at the end, has turned out to be a crucial feature. The students have to present their design and concept to the customer at the beginning and to make a formal contract concerning functionalities, milestones, and deadlines. They have to organize both their work and themselves in a way that ensures the final success of the project. Of course, there are accompanying courses in software engineering or project management – but nevertheless, there is an intended high degree of "learning by doing" in this concept.

As a topic for the first run of this format at TUM, we chose molecular dynamics for nanofluidic simulations. There were several reasons for this decision. First, nanofluidics is a very interesting and active field of research; second, particle methods such as molecular dynamics are not in the primary focus of the CSE program and are, hence, a nice supplement to the program; third, the underlying mathematical models as well as the numerical algorithms needed are not too complicated, compared with an implementation of a Navier Stokes solver,

for example; finally, our group is involved in a research project dealing with nanofluidic simulations of condensation processes in real fluids on the molecular level, which provides a nice environment for this student project.

The remainder of this paper is organized as follows: In Sect. 2, we briefly present the necessary fundamentals of molecular dynamics simulations for nanofluidics. Then, in Sect. 3, the issue of parallel computing – as far as relevant for this student project – is addressed. Afterwards, Sect. 4 is dedicated to the outline of the student project *MolDyn* itself. Finally, in Sect. 5, the current state of the still running project as well as outcomes and future perspectives are discussed.

2 Molecular Dynamics Basics

The classical Molecular Dynamics (MD) simulation is well-established for the simulation of fluids on an atomic scale [7, 8, 9]. For real simple fluids, a molecule is modelled as a set of atoms or sites, resp. Molecules are rigid, which means that there are no internal degrees of freedom and only non-bonded interactions have to be considered. The student project has to implement a single site type only: the well-known Lennard-Jones (LJ) sphere. Sites interact with each other and there's a potential associated to this interaction. Only binary interactions and hence pair potentials are considered.

The LJ 12-6 potential $u_{ij}(r) = 4\varepsilon \left[(\sigma/r_{ij})^{12} - (\sigma/r_{ij})^{6} \right]$ for two LJ centers i and j with positions $\boldsymbol{r}_i$, $\boldsymbol{r}_j$ and distance $r_{ij} := \left| \boldsymbol{r}_j - \boldsymbol{r}_i \right|$ is a semi-empiric function. It covers repulsion through the empiric r_{ij}^{-12} term and dispersive or van der Waals attraction, resp., through the physically based r_{ij}^{-6} term. Therefore, it can be used to model the intermolecular interactions of non-polar or weakly polar fluids. The reference length parameter σ corresponding to an atom diameter and the energy parameter ε are adjustable parameters. For mixtures, the modified Lorentz-Berthelot rules provide these for the interaction between molecules of different components A and B: $\sigma_{AB} := (\sigma_A + \sigma_B)/2$ and $\varepsilon_{AB} := \xi\sqrt{\varepsilon_a \varepsilon_B}$. The parameter ξ is heuristic and adjustable with a value close to 1 [10]. The force between two sites is given by the gradient $\boldsymbol{F}_{ij} = -\nabla u_{ij}$ and fulfils Newton's third law (*actio = reactio*). Typically, Hamiltonian mechanics is used, and the phasespace consists of all molecule positions r_i and velocities v_i. Newton's equations of motion for the atoms set up a system of ordinary differential equations: $\dot{\boldsymbol{v}}_i = \sum_{j \neq i} \boldsymbol{F}_{ij}/m_i$. The resulting initial value problem is solved numerically with some suitable time integration scheme. Regarding a canonical ensemble, the number of molecules N, the volume V, and the temperature T are kept constant. The cuboid domain with periodic boundary conditions is used and during the simulation run, the temperature is controlled with a thermostat. This is achieved through a simple isokinetic scaling of the velocities here. Regarding multicentered molecules, the phasespace also has to contain the orientations and angular velocities. The intermolecular force between two molecules is the sum of all site-site forces, which also generate torque. An enhanced time integration scheme takes care of that [11].

3 Parallel Algorithms

Due to the enormous computational requirements resulting from the large number of molecules to simulate and the small time steps, MD still abuts against the limits of today's possibilities even on supercomputers.

First, the sequential algorithm has to be optimized. Due to the property of the used potential and force to decay very fast with increasing distance, there are typically many very small values to be found in the force matrix, which corresponds to the interaction graph. Considering only interactions of molecules below a certain distance, the cut-off radius r_c, these values will be neglected and set to zero, and the matrix will get sparse. To avoid unneccessary distance calculations, the linked cell method is applied. It, first, bins all molecules using regular cells and then only examines molecules within a cell and neighbouring cells within the cut-off radius. The cells approximate the sphere-shaped interaction volume, and Newton's third law can be incorporated, including only half of the neighbours and adding/subtracting a calculated force to both molecules at once. The algorithm achieves linear complexity for short-range MD fluid simulations and provides a basis for parallel versions.

The target platform are clusters of workstations, which belong to the class of distributed memory (NORMA) machines. There are three common parallelisation strategies known for parallel MD simulations [12]: Replicated Data (RD), Force Decomposition (FD), and Spatial Decomposition (SD).

Replicated Data, which is also known as Atom Decomposition, is the easiest, but least efficient strategy: each Processing Element (PE) will store all data, but calculates only one part of the molecules without taking into account their positions. In contrast to a shared memory parallelisation where all the data can be accessed from every PE by definition and are stored only once, the redundant replicated data has to be synchronised. The communication-computation ratio is usually quite large for massively parallel systems, since each PE has to communicate with all other PEs.

Force Decomposition not only distributes the molecules to update among the PEs but also assigns parts of the force matrix to each PE (cf. Fig. 1), where each PE calculates only partial forces of its molecules. Rearranging the columns reduces the communication partners for each PE. Regarding e.g. PE 7 in Fig. 1, molecule positions are needed from PEs 3,11,15,5,6,8 and partial sums have to be exchanged with 5,6,8. Each PE communicates with $O(\sqrt{p})$ PEs, where p denotes the number of PEs. But this is depending on the number itself. A square number is favourable whereas a prime number leads to the RD approach.

Spatial Decomposition is used for the parallelisation of various applications. The whole domain is divided into subdomains and distributed among the PEs. Regarding MD, each PE calculates the associated molecules. Special care has to be taken regarding interactions between two molecules belonging to different PEs. These occur only at the boundaries, and the molecules concerned have to be replicated and synchronized. This is equivalent to enlarging each subdomain with an overlapping halo region. A heterogeneous distribution of molecules entails unequal molecule quantities for the PEs and requires load balancing.

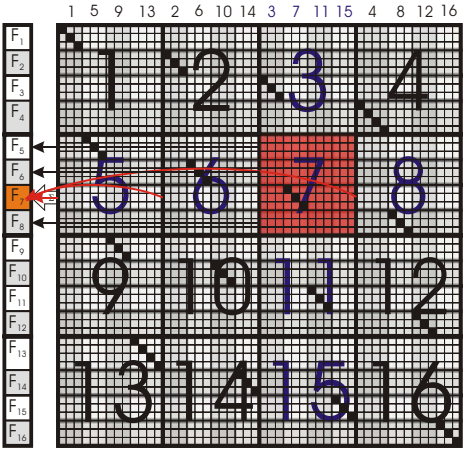


Fig. 1. Force Decomposition for 16 PEs (The force matrix calculation is distributed among the PEs, which have to communicate molecule positions and partial sums.)

4 The *MolDyn* Project

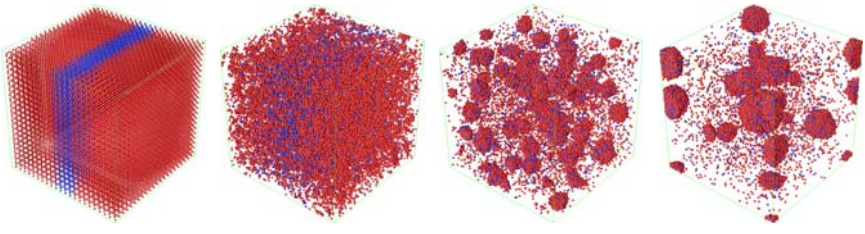
4.1 Task

The student project is about a software system to perform molecular dynamics simulations and visualise the results. The simulations have to deal with mixtures of rigid multiple sites molecules. To organise the molecule types used, the component definitions are stored in an extensible library. Tools shall help the user with the set up and administration. There's also a need for utilities to generate the necessary input data for multicomponent simulations from macroscopic parameters, as mentioned in the first example of Sect. 4.2. Since a simulation run typically takes a long time, an interruption has to be possible, whereas the program will save all necessary data to a file needed for a restart. Platform independence is required to some extent to run the simulation on various supercomputing environments offering MPI communication and also on forthcoming systems, since the rapid product cycle for computer hardware is expected to be substantially shorter than for the simulation software. The targeted user is an experienced researcher. It has to be considered that the program structure must be flexible and should allow to include innovations such as new site types or evaluation functions in the future. Due to the large computational requirements, the modularity has to be combined with efficiency, which refers to the chosen algorithms as well as to implementation issues. Documentation is always important, but particulary if the code shall serve as a base for further development by the customer.

It's not only the result that counts, but also the fulfilment of conditions defined within the contract. Besides the detailed system requirements, which have to be negotiated with the customer, mainly time restrictions have to be obeyed. A time plan including milestones is mandatory and part of the contract.

4.2 Test Cases

The simulation package has to be capable to simulate the dynamics of multicentered molecules. It particularly has to pass the given test cases, presented next. Fig. 2 shows an example of a canonical ensemble of $N = 40000$ atoms in a domain of volume $V^* = 97^3$ with periodic boundary conditions at temperature $T^* = 0.7$. A face-centered cubic lattice serves as the initial configuration (cf. Fig. 2a). The observable nucleation process initiates a phase transition, which also has consequences for the computational effort needed. There is another NVT-ensemble test case, where spherical methane and ethane clusters of molecules in a domain of volume $V^* = 120^3$ with periodic boundary conditions at a temperature $T^* = 1.0739$ are colliding. The vaporisation to the surrounding vacuum and the collision will spread the molecules over the domain, but at the beginning the clusters cause an inhomogeneous distribution over the domain and a relatively large amount of interactions to be calculated. These test cases define the functionality of the code and constitute a requirement for the acceptance of the final product.



(a) $t^* = 0$ **(b)** $t^* = 100$ **(c)** $t^* = 500$ **(d)** $t^* = 3000$

Fig. 2. Nucleation process

4.3 Related Courses

Of course, although there is a strong component of "learning by doing" in this format, the participating students are not left alone – concerning neither MD nor the software and project issues. For example, the customer provided a detailed description of the product expected before the project started, and there was a block course on MD tailored to the project's needs as well as to the students' knowledge and held esp. for the participants of this project. Furthermore, as a part of the regular CSE course program, there is an accompanying lecture on software engineering, which provides the basic background for this part of the project. Hence, the student project is well embedded into the CSE and BGCE programs at TUM.

5 Current State and Outcomes

The first action of the new team was to self-organize. Each of the five members got his/her own role with corresponding responsibilities, such as project manager, configuration manager, quality assurance, development, or documentation.

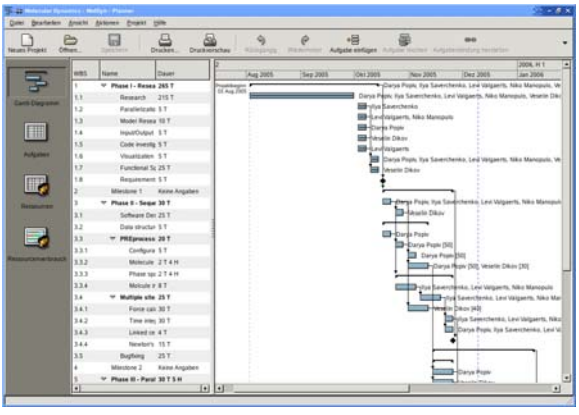


Fig. 3. Time planning with Gantt-diagram

The team also had to agree upon a common environment in terms of tools to be used. Besides a document processor and a project management system also tools such as an IDE and a CVS had to be chosen. The students also had to get acquainted with MD and MD implementation details, which was further supported by the advisor and additional literature. Before the design and implementation phase, the requirements and functional specification was written. Customer meetings to clarify the requirements were supplemented by electronical communication via email for renegotiation.

An important issue for the accomplishment of a project is time management. The students had to set up a work plan. Working packages to carry out the different tasks had to be identified and associated to a time schedule with related milestones. The first of these mandatory milestones included the work plan itself. Fig. 3 shows a Gantt-diagram created by the students with some work packages assigned to the team members.

The simulation was divided into the classical parts: preprocessing, computation, and postprocessing. The preprocessing programs dealing with the XML and binary input files are written in Java and responsible to set up an initial configuration for the phasespace to start a simulation. The team was free to use adequate programming languages for different tasks and decided to use C++ for the parallel simulation code. The simulation results have to be evaluated, and a visualisation of the molecules is required. The open-source visualisation framework OpenDX was integrated here to save work. Recently, the second milestone was obtained and a sequential version of the program, including the interface to and customization of the visualisation, was delivered to the customer. Right now, the students struggle with parallelisation issues and work on the implementation of the FD method combined with the linked cells algorithm.

From the very beginning, all five participants have been enthusiastic and highly motivated. Although they all have a bachelor's degree in engineering or science and, hence, more a CSE than a computer science background, they are esp. interested in the software engineering aspects. They organized the project

and themselves in a very disciplined way, and they now really experience that applying software engineering principles is not an overhead not worthwhile, but is inevitable for larger projects and, then, really pays off. At the beginning, the original time schedule had to be adapted several times (mainly due to the fact that their MD background was close to zero). Nevertheless, after the initial phase, all deadlines were met, and the team certainly will be able to deliver a quite impressive product at the end. Since the students consider this project to be one of the most challenging, but also most interesting parts of their master's studies, we will definitely go on with this format.

References

1. D. E. Post and L. G. Votta. *Computational Science Demands a New Paradigm.* Physics Today, 2005 (no. 1), pp. 35-41.
2. M. R. Benioff and E. D. Lazowska. *Computational Science: Ensuring America's Competitiveness.* PITAC report, 2005.
3. Technische Universität München. *International Master's Program CSE – Computational Science and Engineering.* `http://www.cse.tum.de/`.
4. J. Ludewig and R. Reißing: *Teaching what they need instead of teaching what we like – the new Software Engineering curriculum at the Universität Stuttgart.* Information and Software Technology 40 (4), 1998, pp. 239-244.
5. M. Bernreuther and H.-J. Bungartz: *Experiences with Software Projects for CSE Education.* Submitted to Computing in Science & Engineering.
6. Technische Universität München, Friedrich-Alexander-Universität Erlangen-Nürnberg. *BGCE – The Bavarian Graduate School of Computational Engineering.* `http://www.bgce.de`.
7. M.P. Allen and D.J. Tildesley: *Computer Simulation of Liquids.* Oxford University Press, 2003 (reprint).
8. R.J. Sadus: *Molecular Simulation of Fluids - Theory, Algorithms and Object-Orientation.* Elsevier, 1999.
9. D. Frenkel and B. Smit: *Understanding Molecular Simulation - From Algorithms to Applications.* Academic Press, 2002 (2nd ed.).
10. J. Stoll, J. Vrabec and H. Hasse: *Vapor-Liquid Equilibria of Mixtures Containing Nitrogen, Oxygen, Carbon Dioxide, and Ethane.* AIChE J. 49: 2003, 2187–2198.
11. D. Fincham: *Leapfrog rotational algorithms.* Molec. Sim. 8 1992, 165–178.
12. S. Plimpton: *Fast parallel algorithms for short-range molecular dynamics.* J. Comp. Phys. 117 1995, 1–19.