Scientific Workflows with XMDD: A Way to Use Process Modeling in Computational Science Education

Anna-Lena Lamprecht¹ and Tiziana Margaria¹,²

¹ University of Potsdam, Institute for Informatics and Computational Science, Potsdam, Germany
lamprecht@cs.uni-potsdam.de
² University of Limerick and LERO - The Irish Software Research Centre, Limerick, Ireland
tiziana.margaria@lero.ie

Abstract
Process models are well suited to describe in a formal but still intuitive fashion what a system should do. They can thus play a central role in problem-based computational science education with regard to qualifying students for the design and implementation of software applications for their specific needs without putting the focus on the technical part of coding. eXtreme Model Driven Design (XMDD) is a software development paradigm that explicitly focuses on the What (solving problems) rather than on the How (the technical skills of writing code). In this paper we describe how we apply an XMDD-based process modeling and execution framework for scientific workflow projects in the scope of a computer science course for students with a background in natural sciences.

Keywords: process modeling, scientific workflows, computational science, model-driven development

1 Introduction and Motivation

Computer Science institutes are nowadays not only responsible for teaching courses for their own students, but increasingly also for introductory courses to students from other institutes, and specialized courses for other study programs, especially in two-discipline Master programs. The course on scientific workflows that we address in this paper is an example of the latter. It is aligned to the technical background and requirements of Master students of interdisciplinary programs that include some share of applied computer science. Our students are enrolled in the Master programs in Bioinformatics and in Geovisualization. They already hold a Bachelor’s degree in a natural science discipline (such as biology or geography), and can be regarded as application experts in their respective scientific domain, but they have not been trained in computer science or programming yet. We taught this "Computer Science for Natural Scientists" course at the University of Potsdam, Germany, for several years now. It actually consists of two components: a rather traditional introductory computer science course (data representation and data structures, principles of algorithms, the computer architecture, some programming
basics, the purpose of object orientation and a brief introduction to databases), and a second component specifically about process modeling and scientific workflows.

Our principal goal is to empower students with little or no background in Computer Science or in computation to autonomously design and implement software for their specific computational problems, which are different forms of scientific workflows. The design phase analyzes the problem and develops a corresponding logical model: be it a program model or just a design of the process, it may only exist in the subject’s head, but it is always there. Design is the problem solving part, immersed in the application domain space. In the implementation phase, the design is translated into actually executable code. This task requires in general coding skills, and is thus difficult to achieve with little knowledge. However, in modern process modeling frameworks, the (in most cases graphical) models are domain-specific, the single domain-specific elements (algorithms, data access, data manipulation, format transformations, communication and presentation) often preexist and are often directly executable, so that once the process is composed, no manual coding is required any more to obtain an executable program. With XMDD we are indeed in this situation. Accordingly, we can focus on the creative, rewarding, and problem-specific design phase, and teach the students how to get from their particular (computational) problem to a comprehensive program model that describes how this problem can be solved computationally.

We see these skills as an extremely powerful complement to the "traditional" part of the course: While it is essential for everyone working in software development to understand basics like data types and control structures, a one or two semesters course is not enough for the students to become skilled programmers. Through the process modeling approach they learn to use a both syntactically and technologically easier way to implement custom software applications. While the development of algorithms and of Big Data analysis execution engines will remain in the hands of specialized experts, the organization of such capabilities to solve a problem is an essential skill for any researcher today. Indeed, we are now cooperating within Lero with ICHEC, the Irish Centre for High-End Computing, to create an even better platform that integrates and combines both worlds, making high-end computation design more readily available to researchers without computational science skills.

Didactically, the course follows a learn-by-doing approach: upfront, the participants receive only the minimally required amount of theoretical input. We instead emphasize experimentation and the concrete project work. In this, we follow many didactic ideas typically associated with problem-based learning (cf. [1, 2, 13]), and help the students to develop flexible knowledge, effective problem solving skills and intrinsic motivation, as well as to collect rich practical experience with an own project carried out to the running end-product.

Technically, for the project work we adopt the jABC [15] as process modeling and execution framework. jABC provides these participants with a process management framework that hides low-level details in a service-oriented fashion [10], integrates high-level modeling in the overall development process in a way that user-level models become directly executable [9, 7], and supports ad-hoc adaptations and evolution [6, 8]. The most attractive feature is its simple and intuitive graphical user interface, essential to make it suitable for these educational purposes.

Figure 1 gives an impression of the jABC in action: The process model (called Service Logic Graph, or SLG) on the canvas has been created using workflow building blocks (called Service-Independent Building Blocks, or SIBs) from the library (displayed in the upper left of the window) in a drag&drop fashion, and connecting them with labeled branches representing the flow of control. Once the parameters of the SIBs have been configured (in the SIB inspector in the lower left), the workflow is ready for execution. The control panel of the Tracer plugin (small window in the upper right corner) steers the execution of the models. We see that it is
Figure 1: Process modeling and execution with the jABC framework.

currently executing a SIB: the green-colored branches of the model on the canvas visualize where the execution has currently arrived. The third window in the figure shows an (intermediate) result from the workflow execution; it has been opened by the currently executed SIB.

The jABC is furthermore the current reference implementation of the XMDD (eXtreme Model-Driven Development) paradigm [9] and the One-Thing Approach (OTA) [7], which enforce the rigorous use of user-level models and refinement throughout the software development process and software life cycle. Following the XMDD paradigm, the jABC in fact allows us to begin with the software development at the student’s level of expertise, discussing about the things in their domain and talking about types and functionalities. This enables them to develop the workflows in a prototype based agile fashion working on the models, in a kind of incremental formalization (cf. [14]). The initial processes are first modeled using only so-called Prototype SIBs, which are configurable placeholders for arbitrary functionality. These early iterations capture the logic behind the scientific workflow, in a similar fashion to the abstract and coarse grained reference processes used in business process modeling. They are then successively refined and enriched (replacing the Prototypes by fully implemented SIBs or other SLGs) until an actually executable process model is reached.

The full, professional version of jABC has a wealth of additional features and capabilities including model checking, code generation, workflow synthesis and plugin generation, but for the purpose of the course addressed in this paper the above is indeed sufficient. Given that application definition happens by service composition on this canvas, this is all the students need to master. It can be easily taught and learned in a couple of hours.

Already in the first year the results of the project work reached a very proficient level. In fact, the resulting collection of scientific workflow applications provided rich insights and experiences about the modeling of scientific workflows from the users’ point of view as well as from the computer-science perspective. This encouraged us to design a text book based on these projects, which were considered by domain experts to be suitable to act as a primer for practitioners who wish to learn how to think about domain-specific processes in terms of services and workflows. The result [4] includes an extensive introduction to service- and process-oriented thinking about scientific workflows and corresponding technology, and the
collection of the students’ projects presented in individual chapters as a gallery of applications. Complementing the technical focus of the book, this paper presents the course and its outcomes from a didactic perspective.

The remainder of this paper is structured as follows: Section 2 explains in detail how the course is organized and what happens in the individual units of the course, then Section 3 then discusses the learning outcomes for the students, Section 4 takes a retrospective view of three (almost four) iterations of the course, and Section 5 concludes the paper with the discussion of some further aspects and some ideas for the future.

2 Course Organization

Figure 2 shows the course organization: It comprises course time at the university with weekly meetings, with lectures (the blue squares) or where the lecturers are simply available for individual consultation (green), as well as individual time for working on assignments and on the project (yellow). The red squares indicate the presentations and project results delivery, relevant for the final grading. The four phases of the course are described in the following along the contents and goals. While the course can easily be carried out within one single semester, in the last years we held Phase I as a small add-on to the introductory programming course in the Winter term (Oct.-Feb.), and Phases II – IV took place in the Summer (April-July).

2.1 Phase I: Introduction

The purpose of Phase I is to familiarize the students with process- and service-oriented thinking, in particular in the context of scientific data analysis tasks.

Lecture 1 introduces the field of scientific workflows: using a number of workflow examples from different scientific domains and from different sources, it gives an impression of the breadth of the field. It then develops characteristics of scientific workflow applications and discusses the formal definition proposed by Qin and Fahringer [12]. After looking at the "Scientific Workflow

<table>
<thead>
<tr>
<th>Course Time</th>
<th>Individual Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase I</td>
<td></td>
</tr>
<tr>
<td>Opening Session / Lecture 1</td>
<td>Homework 1</td>
</tr>
<tr>
<td>Lecture 2</td>
<td></td>
</tr>
<tr>
<td>Consultation</td>
<td></td>
</tr>
<tr>
<td>Phase II</td>
<td></td>
</tr>
<tr>
<td>Lecture 3</td>
<td>Homework 2</td>
</tr>
<tr>
<td>Lecture 4</td>
<td></td>
</tr>
<tr>
<td>Consultation</td>
<td></td>
</tr>
<tr>
<td>Phase III</td>
<td></td>
</tr>
<tr>
<td>Short Presentations of Project Ideas</td>
<td>Project work</td>
</tr>
<tr>
<td>Consultation</td>
<td></td>
</tr>
<tr>
<td>Phase IV</td>
<td></td>
</tr>
<tr>
<td>Closing Session</td>
<td>Project &amp; Report Submission</td>
</tr>
</tbody>
</table>

Figure 2: Course organization.
Life Cycle” as described by Ludäscher et al. [5], it focuses on workflow management systems for scientific applications, giving some examples and discussing the conceptual differences between control-flow and data-flow modeling. In the light of workflow management systems as platforms for orchestrating distributed services, the first lecture closes with an introduction to the notion of service orientation and the basic concepts of Web Services. This is all the theory.

Lecture 2 already enters the learn-by-doing domain explaining the basic features of the jABC framework and enables the students to work with it for the first homework. It gives a user-focused introduction to the concept of the SIBs (the workflow building blocks), the available SIB libraries and how to construct workflows. It also shows how to execute and debug jABC process models with the Tracer (the built-in model interpreter) and introduces the idea of the ExecutionContext, a kind of shared memory that jABC process models use for data exchange.

Homework 1 consists of three introductory exercises that complement the lectures to achieve the goal of Phase I: familiarize the students with process- and service-oriented thinking. The first exercise is a “jABC Warmup” assignment, in which the students install themselves the jABC framework, read the jABC basics tutorial, and create their first two process models: one that describes how to prepare the student’s favorite dish using only the Prototype SIB, and one that uses GUI SIBs to realize a quiz consisting of three questions with four possible answers each. The second exercise is already on domain-specific workflows: the students search the web and literature search for typical (computational) workflows in their domain, select an example and describe it as jABC model. The third exercise is on domain-specific services: services from the student’s scientific domain are to be listed and briefly described from a technical perspective. The homework is reviewed and commented by the lecturers and handed back to the students.

2.2 Phase II: Development of a Project Proposal

Phase II guides the students in developing the ideas and proposals for their projects: they identify and select a particular computational science process, analyze and decompose it into basic steps, investigate the availability of suitable services for its realization, and finally produce a written project proposal and a short presentation.

Lecture 3 resumes the discussion of services for workflows started in Lecture 1 with a focus on the project work. It emphasizes again which kinds of software can be used as services in workflows, goes deeper into the technicalities of using SOAP-based and REST-style web services, and presents a variety of concrete service examples that might be useful for the projects.

Lecture 4 resumes Lecture 2 and discusses the more advanced jABC aspects that are required for the project realization, such as the data-flow modeling with the ExecutionContext and the different kinds of SIB parameters, ExecutionContext scopes and hierarchical modeling.

In Homework 2 the students develop a concrete project idea and write a short project proposal. Since they already hold a Bachelor’s degree in a scientific field, they easily find a suitable relevant topic. If not, we help out with suggestions. The proposal contains a short description of the planned computation, a prototypical jABC model illustrating how it works “in principle”, and a list of the required services along with details on their availability and expected integration efforts. Also this homework is reviewed and commented by the lecturers.

2.3 Phase III: Project Realization

This is the main part of the course, starting with 5-10 min (depending on the available time and the number of projects) flash presentations of the project ideas. This way the students get an impression of each other’s project plans, they can exchange direct feedback, ask and answer questions, and note similarities for future exchange, discussion, and mutual help. The
project work then runs in a self-organized way. Weekly consultations are offered by the lecturers during the class time, and are open to all the students, especially when they need assistance. The length of this phase should suit the complexity of the projects and the time required by the students to realize them.

Instead of full-length project presentations, at the end of the lecture time there is a project demonstration session. On this date, the students bring their (almost) completed workflow projects and give a practical demonstration. This session is organized like a fair: everyone can simply visit the others to have a look and try out their results. This demonstration session to the entire class was introduced at request of the students: they asked for another “checkpoint” towards the end of the semester, helping them not to neglect their project work.

2.4 Phase IV: Submission, Evaluation and Grading

Finally, in a closing session remaining open points are discussed, and the details of the project grading are explained, and the formal course evaluation by the students takes place. Usually the submission of the final project results is 2-4 weeks after the last lecture, for further improvements. In addition to the actual jABC project, the students submit a written report (10-20 pages) structured according to a template that foresees four sections (Introduction/Workflow Scenario, Service Analysis, Workflow Realization, and Conclusion). The final grading considers the presentation (10%), the jABC project (40%) and the written project report (50%). The homework and the project demonstration are not graded, but required for passing the course.

3 Learning Outcomes

In terms of competence fields, we describe the skills that the students acquire as follows:

Professional competencies:

- Graduates are familiar with the principles of scientific workflows, process modeling and service orientation.

- They are able to identify services suitable for use in workflow applications, and to distinguish between different kinds of service interfaces and the ways to access them.

- They are trained in using the jABC framework for process development and execution. They are familiar with the jABC’s standard SIB library and can integrate command-line tools and REST-style services on their own.

Methodological competencies:

- Graduates are able to analyze computational problems from their scientific domain from an informatics perspective. They can decompose them into individual steps, and identify and describe the process flow connecting them.

- They are capable of searching the internet and relevant publications for services providing the functionalities for the individual steps.

- They can run a software project, and finally provide a tested release with adequate documentation.

- They are able to communicate their ideas and the results of their project work both orally and in written form.
Action competencies:

- Graduates are able to design and (depending on service availability) to implement software applications for their specific computational problems.
- They have internalized the importance of reuse in software development, from the perspective of benefiting from existing resources in own applications as well as with regard to making own resources available for reuse.
- They are prepared for real-world problems that typically occur in IT projects when working with existing services and resources.

Especially the action competencies would be far less developed if we would let the students simply work on predefined small-step process modeling exercises. The practical workflow project, where they choose their own target computational problem, is all about action. Of course typically it gets quite demanding for both students and lecturers; however, it is so effective that it is worth the effort. The project provides a “guided reality check”, resembling likely situations of their future working lives, but with the lecturers’ assistance. The typical problems are either related to the concrete workflow realization in jABC, or they concern more general issues like data type incompatibilities, unreliable and switched off remote services, or data and software licensing issues. In extreme cases, issues in the context can mean that a project can not be realized as planned, at least not within the given time. This may lead to re-planning and re-targeting, from which the students learn a lot. Good grades are connected with convincing and competent explanations of the problems in their reports. In the normal case, however, students terminate the course with a sense of achievement, having solved a computational problem from their scientific domain in a fashion that is ahead of the state-of-the-art.

4 Retrospective View

We taught first versions of this course at the University of Potsdam in the academic years 2008/09 and 2010/11. Since 2011/12 it took place each year in the format described here, and we are currently in its 4th iteration.

Table 1 surveys course attendance numbers and size and complexity of the developed projects over the last three years. While the total number of students attending the course varies in accordance with the matriculation numbers of the study programs, there have always been more students from geovisualization than from bioinformatics. In 2011/12, with only 17 students attending the course, we let each student work individually on a project. In the other years,
with clearly higher attendance numbers, we encouraged the students to work on the workflow projects in pairs or groups of three. This was on the one hand necessary to ensure that our supervision capacities would be sufficient for all projects. On the other hand, teamwork is favorable for projects of this kind, and it helps the students to develop effective communication and collaboration skills in the area of software development.

The key figures do not show significant differences between the projects. Measured in terms of the number of used SIBs, the average project size does not vary widely. The average number of models per project has slightly increased over the years. What the numbers show is that the projects are of moderate, but not trivial size. We measured the average model complexity in terms of the McCabe number [11], which gives the number of linearly independent paths through a process model and is a commonly used quantitative complexity measure for software. Usually, 10 is considered to be the upper acceptable limit of McCabe complexity. Hence, although the average McCabe number of the models varies between the years, it is always clearly below this threshold, indicating that they are at a comprehensible level.

Thematically, the projects come from different areas of bioinformatics and geovisualization, and sometimes also tackle interdisciplinary questions. Most projects realize data analysis workflows, which are typically organized in three phases: (1) data collection or loading, (2) data (pre-) processing and analysis, and (3) result production and visualization. The input data is often loaded from some locally available sources or provided by the user, but in many cases also dynamically fetched from public repositories like the EMBL databases, which provide web service interfaces for programmatic access. The data analysis part is carried out by domain-specific tools and services. Popular examples for bioinformatics are the EMBOSS tools, the EBI web services and custom GNU R scripts, while in geovisualization the Generic Mapping Tools, the CSISS Web services and the Google Maps REST API are frequently used. Result production then usually means to aggregate and store the obtained results in some textual and/or graphical format that can easily be displayed.

A detailed description and analysis of the individual projects is out of the scope of this paper. A collection of concrete project examples and some more statistics are available in [4]. All these projects are also interesting research objects for us as process modeling researchers, and we are currently working on a larger empirical analysis of jABC projects from different domains using the jABCstats tools [16].

Because our process modeling course is part of a larger course, unfortunately we do not have any official course evaluations available specifically for this part. In particular it is principally difficult to systematically assess to which extent the more abstract learning outcomes have actually been reached. We were however reported several "success stories" from students: they were facing similar problems later during their studies or at their jobs, and remembered and used the approaches and techniques learned in this course. In terms of grades, Figure 3 shows that most students in fact achieve good (grades 1.7, 2.0, 2.3 in the German system, a B level) or very good (grades 1.0 and 1.3, A level) results. Interestingly, the median of the given grades is stable over the years, but the range has become smaller. We will investigate possible reasons for this effect in the future. Only a few students did not pass the course (grade 5.0). The reason was however not insufficient performance in the project, but simply that the students decided to quit the course during the semester and hence did not submit anything at the end of the term. This is often the case, especially at the beginning of a program as in this course, because there is no penalty for this in the German system. According to our experience, the workload for the students corresponds to the 3 ECTS credits they receive when they complete the course, that is, it comprises more or less 90 working hours.
Figure 3: Grade distribution (2012-2014)

5 Conclusion

We believe that process modeling can and should play a central role in computational science education, and is very effective to qualify students for the design and implementation of software applications for their specific needs. In this paper we have reported on our experiences with the use workflow projects carried out with the jABC process modeling framework for teaching service-oriented software development to students from bioinformatics and geovisualization Master programs at the University of Potsdam. Our course aims at familiarizing with scientific workflows non-CS students that have a significant prior background in a scientific field. We achieve it by linking an IT project with a high-end computational content to their experience and domain knowledge, in a learn-by-doing fashion. Co-education of students from different programs was not a problem at all. While the number of institutions offering courses on the various aspects of scientific workflows has seen a steady increase over the last years, we are not aware of any other course that follows our rigorous project-oriented approach.

We have meanwhile developed a 6 ECTS variant of the course, called “Process Modeling in Scientific Applications”, that serves the converse purpose: to teach the basics of scientific workflows this time to Bachelor students in the first year of Computer Science and Computational Science. These students are in the process of acquiring solid programming and software development skills, but they have hardly any background in other scientific disciplines. The course setup is very similar, with minor adaptations that better align it with the knowledge of this audience. These students appreciate in the first place an opportunity to get in contact with scientific applications. While in the course of the natural scientists assistance is needed to carry out programming tasks (for instance providing SIBs to access SOAP-based web services), here the students are able to solve these kinds of problems themselves. They need however support and assistance in the application domain. The experience from the first and so far only run of this course, with 27 students working on 15 projects, is very positive. They liked the possibility to work on a project that they could choose based on their personal interests, and accordingly finished the course with good results.
In the future we envisage to offer a course that brings both groups together, with project work carried out in interdisciplinary pairs: a student of a natural science subject and a CS partner. Taking on the roles of the “domain expert” and the “programmer”, respectively, they would work together on one problem - from different perspectives, and united by their process model(s). This setting would comply with the vision of interdisciplinary collaboration with domain experts that many developers of process and workflow management systems share.

Furthermore, we are designing advanced courses in scientific workflow modeling and design, that address also questions of semantic domain modeling and the use of formal methods like model checking and synthesis techniques along the lines described in [3]. We would like to maintain also there the clear project-oriented approach, but a larger share of more theoretical lectures will most likely be required to thoroughly introduce these concepts and methods.

References