CASE STUDY: UNIVERSITY EXAMPLE

Carlos Sáenz-Adán[1*], Beatriz Pérez[1], Francisco J. García-Izquierdo[1], Luc Moreau[2]

[1]*Dept. of Mathematics and Computer Science, Univ. of La Rioja, La Rioja, Spain,*

{carlos.saenz,beatriz.perez,francisco.garcia}@unirioja.es

[2]*Dept. of Informatics, King's College London, London, UK,*

luc.moreau@kcl.ac.uk

The results presented in this appendix are part of the evaluation we have made to show the feasibility of our proposal UML2PROV presented in the paper [1]. In that paper, we show the evaluation we have made of our proposal by applying it both to a legacy application named GelJ [2] built without UML (*retroactive* scenario) and to an academic application built from a UML design (*proactive* scenario). While in [2] we mainly focus on the GelJ case study because it is more complex, here we describe in detail the application of UML2PROV to the *proactive* scenario, the University example, using the same evaluation aspects we have taken into account in the paper [1]. We would like to note that, in contrast to the GelJ case study, in the academic University example we play the role of software designers, developers, provenance consumers and potential users.

The University case study correspond to an example, slightly modified from [3], related to the enrolment and attendance of students to seminars that are held during a University course. This application mainly allows users to perform three actions in the context of a university. The first allows administrative staff to check if a student can apply for a given seminar. The second offers the possibility of enrolling a student in a seminar. Finally, the third is related to the overall process which encompasses the evaluation of students' performance in such seminars through exams, con-sidering from the time an exam is prepared until the student is informed of her/his mark. It is worth noting that the University application's database only contains information about specific characteristics of the elements conforming the system (e.g., students, seminars, exams, and so on). It does not keep information about the requests for checking if it is possible to apply for a seminar, or for example, the specific process of preparing and taking an exam, and informing about the mark.

In order to give an unbiased evaluation of our approach, based on a representative use of the University system, we have defined an example execution considering several scenarios (see Algorithm 1). Concretely, this execution is divided in two phases: in the first phase (lines from 2 to 9), 25 students ask for enrolling into a seminar (line 4), after the affirmative response, they are enrolled into the seminar (line 6), and finally, the student' performance evaluation begins

by proceeding with an exam (line 7). In the second phase (lines 10-13), another 25 students (already enrolled) proceed with an exam (line 12). Consequently, the defined benchmark involves the execution of the three cited functionalities of the application. The performance of this example execution incurs in the execution of about 1700 operations in the tool's source code.

```
1  seminar ← findSeminar(idSeminar);
2  for i ← 0 to 24 do
3      student ← findStudent(i);
4      allowEnrol ← askStaffForEnrolling (student, seminar);
5      if allowEnrol then
6          enrolStudent (student, seminar);
7          proceedWithExam (student, seminar);
8      end
9  end
10 for i ← 25 to 49 do
11     student ← findStudent(i);
12     proceedWithExam (student, seminar);
13 end
```

Algorithm 1: Example execution algorithm in pseudocode

The evaluation was run in the same personal computer and under the same conditions as in the GelJ case study.

1 Particularities of the application modes in this case study

In Table 1 we show the tasks comprising each application mode for this case study. This information can be used to compare the characteristics of each mode.

Table 1: Overview of tasks in UML2PROV *Application Modes*.

	Case study - University		
Task	**App. Mode 1**	**App. Mode 2**	**App. Mode 3**
T1. Identify the complete CD	–	–	–
T2. Identify provenance requirements	🧍	–	–
T3. Identify classes/operations involved in provenance requirements	🧍	–	–
T4. Discard not identified classes/operations	🧍	–	–
T5. Add stereotypes to selected operations	🧍	–	–
T6. Identify SqDs	🧍	–	–
T7. Design SMDs of selected classes	–	–	–

⏩ Permorfed automatically | 🧍 Requires manual effort | 🧍/⏩ Requires semi-manual effort | – Non executed task

Application Mode 1

Regarding task T2, since in this case study there are no final users interested in the generated provenance, it is not possible to obtain the provenance requirements from them. For this reason, as described previously, we have simulated that we are the final users that raised the provenance questions that serve as requirements. To conduct an unbiased

evaluation, we have not defined the questions from scratch, but we have drawn inspiration from the questions appearing in the First Provenance Challenge [4], adapting them to obtain questions regarding the performance of exams. The resulting questions, depicted in Table 2, represent the provenance requirements (called *provenance use case questions* in PrIMe).

Table 2: Questions identified from Q1 to Q5 about the University case study, together with University classes involved in answering those questions.

ID	Question	Identified Classes
Q1	What is the set of activities that has led an exam as it is?	Exam Teacher Student
Q2	How many answers have an exam?	Exam
Q3	When is the student informed about the mark of an exam?	Exam
Q4	Who has signed the exam?	Exam Student
Q5	What is the date of an exam?	Exam

In this case study, all the UML CD, SqD and SMD diagrams are available at the beginning of the evaluation. Thus, all the performed tasks are tailoring tasks. Neither to reverse engineer the CD (T1) nor to design the SMDs (T7) are required. Similarly, it is not necessary to reverse engineer SqD. We just have to select the SqDs related to the provenance requirements (T6).

Inspired by the second phase of PrIMe, from the UML design we have identified those classes and operations (called *actors* in PrIMe) involved in answering the identified questions (T3), following the same procedure described for GelJ. The result was the identification of 3 classes and 11 operations out of 11 classes and 37 operations that compose University application (i.e., $\sim$27% of the classes and $\sim$29% of the operations of the University application were used. The rest of classes/operations were discarded (T4). These classes are shown in column "Identified Classes" of Table 2. In addition, to obtain more meaningful provenance, we assigned stereotypes to the UML operations in class diagram (T5). As for SqDs, we selected those related to the provenance requirements. This task resulted in a set of 25 messages. Regarding SMDs, we used the UML SMDs for those classes whose states are related to the provenance requirements. This led to 2 SMDs with 7 states, and 9 transitions.

As in the GelJ case study, the greatest effort goes into identifying the provenance requirements (T2) and selecting the classes and operations involved (T3 and T4).

Application Modes 2 and 3

In these modes no tasks are performed, so no effort has been made. More specifically, we have taken the original SqDs and CDs without tailoring them (CDs, in both modes and SqDs, in Mode 2). Concretely, the UML design encompasses a CD with 11 classes and 37 operations, and a set of SqDs with 25 messages in total.

1.1 Analysis

Next, we analyse those evaluation aspects explicitly related to the University case study.

Table 3: Variables evaluated for the considered UML2PROV *Application Modes* in the University case study.

Application Mode (UML diagrams)	No. templates	Total size of templates	No. Variables	No. Set of bindings	Sets of bindings size (MB)	Expanded templates size (MB)	No. executions instrumented operations	% instrumented executed operations	Execution time (ms)	Time overhead
Mode 1 (SqD, SMD, CD)	25	20KB	115	687	1,4	2,3	687	41,79%	656,19	25,92%
Mode 2 (SqD, CD)	63	47KB	269	1.644	2,0	3,3	1.644	100%	802,29	53,96%
Mode 3 (CD)	40	26KB	137	1.644	1,9	2,0	1.644	100%	747,31	43,41%

1.1.1 Aspect 1: Generation of the provenance design

In line with the results obtained for GelJ, the time-cost for generating the templates, a few milliseconds per template, is considered negligible compared to the time-cost of performing this task manually (this information is not depicted in Table 3 because we consider it trivial). As for the implications of the followed mode, the results show that the closer the UML design fits the application provenance requirements, the less templates are generated and consequently, the smaller their total size and the faster their generation. Table 3 shows that Mode 1 (with a tailored UML design), results in the lowest number of templates (25). However, Modes 2 and 3 present a higher number of templates (63 and 40, respectively). These figures confirm that a greater initial effort to more accurately tailor the UML according to a set of provenance requirements, results in fewer templates.

1.1.2 Aspect 2: Instrumentation of the application

The column "No. variables" of Table 3 depicts the number of instructions included in the BGM for bindings generation.

Mode 1, with least UML elements, leads to the BGM with fewest instructions (115) against Modes 2 and 3 that generate BGMs with more instructions within (269 and 137, respectively). Given these results, we can see that although Mode 1 has fewer number of instructions, the difference with Mode 3 is relatively small (115 vs 137 instructions). This small difference is because of three main reasons. First, due to the small difference between the considered operations in Mode 1 (11 operations, see Table 1), and Mode 3 (37 operations). Second, because Mode 1 generates templates from SqDs and SMDs, unlikely Mode 3. Third, since Mode 1 considers CDs with stereotypes, which incurs in templates with more variables.

Nevertheless, these results are in the line of those obtained from GelJ [1]: the effort devoted to more precisely tailor the UML design according to the provenance requirements results in a simplification of the BGM, which has significant implications in the performance.

1.1.3 Aspect 4: Storage and Run-time overhead

The overhead attributable to provenance capture is associated with the number of executions of instrumented operations (see Table 3, where column "No. set of bindings" matches this number). In this case study, Mode 1 generates the least number of set of bindings (687), and consequently, led to the least run-time overhead (25.92%) and storage needs (1.4MB). Conversely, Modes 2 and 3 generated more set of bindings (1,644 both of them), and yielded the more time overhead (53.96% and 43.41%, respectively) and storage needs (2MB and 1.9MB). Mode 2 will always require more storage since it takes into account the whole Class diagram (unlike Mode 1) and additionally, it does not discard the Sequence diagrams (in contrast to Mode 3).

Table 4: For each mode, it is indicated if questions Q1-Q5 of Table 2 can be answered completely (C), sufficiently (S), partially (P) or cannot be answered (N). If a question can be answered, the number of elements of the provenance involved in its answer appears in brackets.

	Q1	Q2	Q3	Q4	Q5
Mode 1	S(5)	S(6)	S(3)	S(3)	S(2)
Mode 2	C(12)	S(6)	N	S(3)	S(2)
Mode 3	C(12)	S(6)	N	S(3)	S(2)

Finally, the difference between column "Set of bindings size" and "Expanded templates size" in Table 3 confirms how the use of the PROV-Template approach for generating provenance reduces the storage requirements for the set of bindings compared to the expanded templates.

1.1.4 Aspect 5: Quality of provenance

In this section we will focus on the quality of the provenance generated from the University example. To analyse this aspect, we study if the collected provenance answers *completely* (C), *sufficiently* (S), *partially* (P) or it cannot answer (N) the questions in Table 2.

Completely When the user indicated that the answer was more detailed that what she/he expected.

Sufficiently When the user indicated that the level of detail was enough.

Partially When the answer did not satisfy the user.

No When it was not possible to answer the question.

Table 4 summarizes our conclusions, showing the number of elements (`prov:Entity`, `prov:Activity`, and `prov:Agent`) involved in such an answer, when it can be responded (i.e., when the response is not classified as N). Based on these results, we identified three kind of implications the mode used to tailor the UML design may have on the answers.

No effect. The followed mode had *no effect* on the ability to answer to questions Q2 and Q5. More specifically, the answer to questions Q2 and Q5 relies upon the values of attributes belonging to the class `Exam`. Since the three modes take into account a CD with all the attributes of class `Exam`, the provenance obtained from can answer these questions.

More detailed information. The answer to Q1 relies upon the operations identified in classes `Exam`, `Teacher`, and `Student`. Mode 1, which only identifies a set of operations, answers Q1 with a sufficient number of elements (5). However, Modes 2 and 3, encompassing the whole operations, answers Q1 with more elements than necessary (12 both of them).

Crucial. The UML diagrams supported by UML2PROV model only certain aspects of an application's behaviour. Consequently, the generated provenance will contain information about these aspects. The answer to question Q3 relies upon information provided by SMDs; thus, Mode 1, which is the only one considering SMDs, is the unique mode capable for answering Q3.

References

[1] C. Sáenz-Adán, B. Pérez, F. J. García-Izquierdo, and L. Moreau, "Integrating Provenance Capture and UML with UML2PROV: Principles and Experience," submitted for publication in IEEE Transactions on Software Engineering.

[2] J. Heras, C. Domínguez, E. Mata, V. Pascual, C. Lozano, C. Torres, and M. Zarazaga, "GelJ – a tool for analyzing DNA fingerprint gel images," *BMC Bioinformatics*, vol. 16, Aug 2015.

[3] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, *UML@Classroom: An Introduction to Object-Oriented Modeling*. Springer Publishing Company, Incorporated, 2015.

[4] L. Moreau et al., "Special issue: The first provenance challenge," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 409–418, Apr. 2008.