This document is published in:

Ravenscroft, A. et al. (eds.) (2012). *21st Century Learning for 21st Century Skills: 7th European Conference of Technology Enhanced Learning, EC-TEL 2012, Saarbrücken, Germany, September 18-21, 2012. Proceedings*. (Lecture Notes in Computer Science, 7563). Springer, 320-333.
DOI: http://dx.doi.org/10.1007/978-3-642-33263-0_25

© 2012 Springer-Verlag Berlin Heidelberg

Key Action Extraction for Learning Analytics

Maren Scheffel[1], Katja Niemann[1], Derick Leony[2], Abelardo Pardo[2], Hans-Christian Schmitz[1], Martin Wolpers[1], and Carlos Delgado Kloos[2]

[1] Fraunhofer Institute for Applied Information Technology FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{maren.scheffel,katja.niemann}@fit.fraunhofer.de, {hans-christian.schmitz,martin.wolpers}@fit.fraunhofer.de
[2] Universidad Carlos III de Madrid, Avenida Universidad 30, E-28911, Leganés (Madrid), Spain
{abel,dleony,cdk}@it.uc3m.es

Abstract. Analogous to keywords describing the important and relevant content of a document we extract key actions from learners' usage data assuming that they represent important and relevant parts of their learning behaviour. These key actions enable the teachers to better understand the dynamics in their classes and the problems that occur while learning. Based on these insights, teachers can intervene directly as well as improve the quality of their learning material and learning design. We test our approach on usage data collected in a large introductory C programming course at a university and discuss the results based on the feedback of the teachers.

Keywords: Usage data, learning analytics, self-regulated learning, activity patterns.

1 Introduction

In order to support students within a course – guiding them when they are on the wrong track, giving advice even when they cannot ask precise questions and general troubleshooting – teachers must be aware of what the students are doing and have been doing so far. In other words: the teachers need information on the students' activities. Such information is also needed for the evaluation of a course, its didactic concept and the materials provided, including contents, tools and tests. Information on the students' activities gives insights into which materials actually have been used and which have not, when troubles occurred, when the students started their work and in which parts of the course they got stuck, etc. Such information can be referred to for optimising the course and thus supporting the learning process.

One can monitor the students' activities and list them in one large file. This file, however, will contain more information than teachers and students can effectively evaluate. It will therefore meet neither the needs of the teachers nor those of the students. This is where learning analytics comes in as Siemens and Long

explain [1]. A distillation of the recorded data is required, so that irrelevant information is filtered out and information overload is avoided. The question now is which means of data distillation are useful and help students and teachers in mastering their tasks.

This paper deals with one particular means of data distillation, namely the extraction of key actions and key action sequences. We claim the hypothesis that key action extraction is a very useful form of data distillation. We prove our hypothesis by implementing the approach in a larger test bed, namely an introductory programming course with theoretical lectures and practical lab sessions. The course was held in the fall semester of 2011 at the Universidad Carlos III de Madrid (UC3M), Spain.

The rest of this paper is structured as follows: in section 2, we will describe background, related work and previous experiments. We will then report on the evaluation of the approach within a larger test bed, namely the already mentioned course on C programming (section 3). Section 4 will deal with the implementation of key action extraction and the setting of parameters for the test bed. In section 5, we will report on the insights the teachers got from the key actions and thus qualitatively evaluate the approach. Finally, in section 6 we will summarise and give an outlook on future work.

2 Related Work and Linguistic Background

2.1 User Monitoring and Learning Analytics

Many university courses consist of self-regulated learning activities. The students take over responsibility for planning and reflecting these activities. Being aware of one's own activities is a prerequisite for reflection. In [2], we have shown that one way of helping a learner to become aware of his actions and learning processes is to record and store his interaction with his computer and to then analyse the collected data. Results of these analyses can on the one hand be used to foster his self-reflection processes or on the other hand to give recommendations, e.g. of further steps in his current learning scenario. The same applies to teachers.

If the number of students in a course is high and the tasks the students are engaged in are not trivial, then the teachers need assistance for keeping track of the students' activities. They cannot constantly observe their students themselves. It will be of great advantage for them if monitoring is supported automatically. A survey conducted by Zinn and Scheuer [3] about the requirements of student tracking tools showed that aspects such as competencies, mastery level of concepts, skills, success rate and frequent mistakes are seen as highly important to teachers. Many teachers said that employing student tracking would allow them (the teachers) to be able to adapt their teaching to the behaviour of the students and to identify problems in the students' learning processes.

Several approaches deal with the creation of feedback for the teacher to enable him to improve the quality of his courses. Kosba et al. [4] for example developed the TADV (Teacher ADVisor) framework which uses data collected by a course management system to build student models and a set of predefined rules to

recognise situations that require teachers' intervention. Based on the student models and the rules, the framework creates advice for the teacher as well as a recommendation for what is best to be sent to the students. Similar to our approach, the goal of this framework is to enable the instructors to improve their feedback and guidance to students. However, our approach does not use predefined rules as we do not want to force the instructor to perform a specific action but to enable him to get new insights into the learning behaviour of his students and thus to rethink and improve his teaching.

The CourseVis system also visualises data taken from a courses management system [5]. It addresses social, cognitive and behavioural aspects of the students' interaction with the course management system. While the discussion graph visualises the number of threads started and their number of follow-ups per student, the discussion plot shows originator, date, topic and number of follow-ups in a scatterplot. The visulisations of the cognitive aspects maps students' performance to the course's concepts and another visualisation deals with the students' access times. While these details can be very useful for teachers, CourseVis only depicts the students' interaction with the course management system and does not take other tool interactions into account.

Another relevant tool is the LOCO-Analyst by Jovanovic et al. [6] which is an educational tool that can be embedded in various LCMSs to analyse the usage data of learners. It has the goal to provide teachers with meaningful feedback about the users' interaction with the learning content. The teacher is for example informed about the average time the students spent on each learning object. When quizzes about the learning content are provided, the teacher gets detailed feedback about the incorrect answers per question and which questions are the toughest ones, as they have an error rate above the average. As with our approach, the LOCO-Analyst does not provide real time help, but tries to help the teacher to improve the quality of his learning content and learning design. Even though the LOCO-Analyst considers a lot of event types for its analysis, it does not create and use sequences of events so far, which we assume give the teachers of a course a better insight into the learning behaviour.

2.2 User Observation and Key Action Extraction

For our approach we make use of the contextualised attention metadata (CAM) schema that allows for describing a user's interaction with digital content [7]. The schema is event centered and is thus well suited for the evaluation and analysis of a user's entire computer usage behaviour. The schema has evolved over years and can be subject to further change in the future.[1] Analysing collected CAM can for example result in an overview of actions taking place in the environment or the discovery of changes, trends, etc. in usage behaviour. In controlled environments, e.g. formal learning, where activities are often scheduled, it can be useful to know what takes place when. In less controlled environments, e.g. informal or blended learning, CAM analyses can help to understand when users are active.

[1] The latest version is available at `https://sites.google.com/site/camschema/`

The concept of a key action is best described in analogy to a key word: a list of key words is a superficial but albeit highly useful semantic representation of a text [8]. The keywords of a text are those content words (or sequences of words) that are significant for the content of the entire text and by which the text can be distinguished from other texts. They do not exhaustively describe what the text is about but still give a clear impression of its theme. Knowing a text's keywords, one can capture the essence of a text's topic and grasp the essential information the text is trying to pass along. In analogy, key actions are those actions that are significant for an underlying set of actions and that give an impression of what has happened. We deem key actions to represent the session they are taken from (with a session being anything from a few minutes or hours up to days, weeks, months, etc.). Key actions (or key action sequences as a parallel to key phrases) indicate what a user has been doing. They give an overview of the essential activities. By no means can they be exhaustive but they provide a superficial yet almost noise-less impression.

Let it be given that we recorded all actions of one student in one practical session. From these we extract the key actions which, so the idea, gives us an impression of what the student essentially did in this session. The approach we apply for the analysis of usage data in order to detect meaningful patterns is the so called n-gram approach [9], followed by a ranking approach, namely *tf*idf* weighting, with *tf* being the term frequency and *idf* the inverse document frequency [10]. The following formula shows how the weight of a word can be calculated in more detail, with $w_{i,j}$ being the weight of word j in document i, $t_{i,j}$ being the term frequency of word j in document i, f_i being the number of documents containing word j and N being the number of documents in the collection: $w_{i,j} = t_{i,j} \cdot log\frac{N}{f_j}$.

We use the *tf*idf* algorithm to weight extracted key actions based on two assumptions: First, if a collection of sessions contains a specific key action more often than another collection of sessions, then this key action is more relevant to the first set of sessions and gets a higher weight. Second, if a key action does not appear as often in the collection of sessions as other actions, it should also get a higher weight in the session it does appear in.

In the fall semester 2010 we ran a first round of tests to extract key actions from CAM collected during a C programming course at UC3M [11]. The collected interactions were gathered from a virtual machine with programming tools and the forum interactions of the learning management system at the time. All calculations were done on the basis of the whole course, i.e. taking all sessions from all students into account at the same time. These first results were deemed useful by the teaching staff and thus some events were analysed in a more detailed way to find frequent error patterns.

The promising results motivated us to continue with our approach. Instead of only analysing the whole course at the same time, we wanted to make the available results more diverse and more detailed. We therefore again deployed our approach during a second year C programming course at the Universidad Carlos III de Madrid. This time, however, we look at the whole course as well

as the different lecture degrees, lab sections, teams and individual students in order to provide teachers with an even better idea of where their students are at and how their learning processes can be supported.

3 Application of Key Action Extraction in Detail

3.1 A C Programming Course

We use the introductory C programming course at the Universidad Carlos III de Madrid as a testbed to test our approach more diversely. The course has four main objectives: The students are supposed to learn to write applications in C with non trivial data structure manipulation and they are told to use industry-level tools such as compiler, debugger, memory profiler and version control to get familiar with it. Additionally, they work in teams and need to create their own plans, divide tasks, solve conflicts etc. Finally, the course is assumed to increase the self-learning capability of the students.

The course is split into two halves. In the first half of the course, the students attend theoretical lectures to get a basic understanding of the programming language and start to program in teams of two in supervised lab sessions. In the second half they are split into larger groups of 4 - 5 people and work together on larger projects. Altogether, 10 instructors work on supervising the course, all of them having at least a master's degree and 4 of them being professors. For the theoretical lecture units, the students are divided into 5 groups that are taught by the professors. For the lab sessions, they are divided into 11 groups that are supervised by the 10 instructors.

At the beginning of the course, the students are offered a Virtual Machine that is already pre-configured. It is a UNIX-based system having the compiler, text editor, debugger and memory profiler – that the students are supposed to use – already installed among additional standard tools, e.g. a browser. Within the Virtual Machine, the main events we consider to be important for further analysis are logged and whenever a student uses the subversion system to download a document or upload own material, the collected events are uploaded and stored in a central database (see [12] and [13] for further details). It is important to mention that the students are well informed about the data collection process and are reminded of it every time they open the Virtual Machine. If they do not want to be monitored, they are able to easily stop it without stopping the use of the Virtual Machine.

For registered students, the course material, i.e. documents, exercises and C files, is accessible via an Apache Server. Every time a page is served, a new log entry is created that can be analysed to extract the events. Additionally, the forum functionalities of the learning platform Moodle are used to offer the students a place to discuss problems and ideas. Given that Moodle offers the teacher the possibility to download all events that took place within a course in Excel format, the forum events can easily be extracted. For the Apache and

Moodle Logs, the students are not able to stop the logging themselves but they are informed about the analysis and can write an e-mail to their teacher stating that they do not want to have their data stored which are then not considered for the analysis.

Altogether, we were able to monitor 33 different event types from three different sources. For the complete course, we collected approximately 340,000 (156,000 first half / 184,000 second half)[2] conducted by 332 distinct students in the period from September 5, 2011, until December 18, 2011.

3.2 Events That Are Monitored

Accessing Web Pages. The course material is accessible to logged in and authorised students. Every time a page is served, a new log entry is written which comprises the time stamp, the user identifier, the IP address, the URL served and the HTTP code. For the following analysis only the successful events are considered, as most of the failed access attempts are due to the reason that a student was not logged in or tried to log in with a false user name or password. Certainly, the students do not only access the course material while learning or programming. They also search for related material, forums that answer their questions or code snippets they can use; additionally, they browse the web for private purposes during breaks. This data is collated using the Virtual Machine that is offered at the beginning of the course and captures the use of the browser as well as of other main tools. Altogether, we were able to collect 131,071 (68,130 / 62,941) web page accesses.

Program, Compile and Debug. The students were told to use the text editor KATE to write their programs as they are supposed to learn to code from scratch in the introductory course without any help from a development environment. Within the Virtual Machine it is stored whenever KATE is opened or closed, but no further information, e.g. about the file that is opened or actions that are performed within KATE, are logged. The students are also expected to learn how to use gdb as debugger to find problems in the code and Valgrind as memory profiler to find memory leaks. As for KATE, it is only stored when the debugger or memory profiler are opened, respectively closed. When the students compile their code using gcc, the compile command is extracted from the command line in the shell including the file that is compiled as well as the resulting warning and error messages. These messages are stored as well, as they keep a lot of insight for the students themselves, e.g. for self-reflection, as well as for the teachers, e.g. to understand with which errors the students get stuck or which warnings they just ignore [11]. With 87,148 (23,298 / 63,850) occurrences, the compile event is the most frequent one in this category, followed by the text editor KATE which was invoked 27,363 (6,875 / 20,488) times. Interestingly, with

[2] If not otherwise indicated, the format of giving the numbers for the whole course followed by the numbers for the first and the second half of the semester in brackets is continued for the rest of this paper.

13,591 (1,081 / 12,510) accesses, the memory profiler is more often used than the debugger that was only accessed 2,838 (583 / 2,255) times.

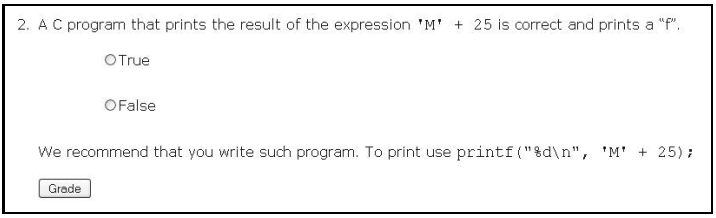

Fig. 1. Screen capture of a Quiz in the Learning Material

Answering Quizzes in the Learning Material. Some of the learning resources contain small quizzes after important sections to support the students in testing their understanding of what they just read, see Figure 1. The students can select an answer and click on "*Grade*" to find out if the answer was correct or not. If the answer was wrong they can try again or display the correct answer. Overall, 21,865 (19,380 / 2,485) times a test was carried out and 5,306 (4,908 / 398) times the students choose to show the correct results after a test and did not try it again.

Communicating in the Course Forum. The Moodle forum used in this course contains 219 discussions which comprise 439 posts. The larger part of these were created during the first half of the semester, namely 148 discussions with 334 posts, while only 71 discussions with 105 posts were created in the second semester. All events conducted in Moodle can be downloaded by the teachers as an Excel file. Each event comprises the name of the course, the time stamp, the IP address, the full name of the student, the event type, e.g. *course view* or *forum add post* and a field for further information, e.g. the name of the viewed course or the title of the post added to the forum. Due to privacy reasons the user names are mapped to the respective user ids before the events are considered for further analyses. In total, there were 52,087 (31,605 / 20,482) LMS-related events. 20,828 (12,886 / 7,942) of them were forum-related.

3.3 Stock Taking: Descriptive Statistics on Tool Usage

The activity of the students is not distributed equally. It is important to take into account that the students work in tandem teams (first half) and small groups (second half) during the lab sessions. Therefore, it is possible that they always work on the computer of one student. However, most students (76,3%) conducted between 100 and 2,000 events. 8,7% conducted less than 100 events. This can have several reasons: (1) they do not use the virtual machine (i.e. configure entirely their personal computer); (2) they use the virtual machine but disable the event recording; (3) they work with somebody else and it is this other person that generates events; (4) they do not actively participate in the

course.[3] The rest of the students (15%) can be said to be very active with 2,000 to 6,507 events per student.

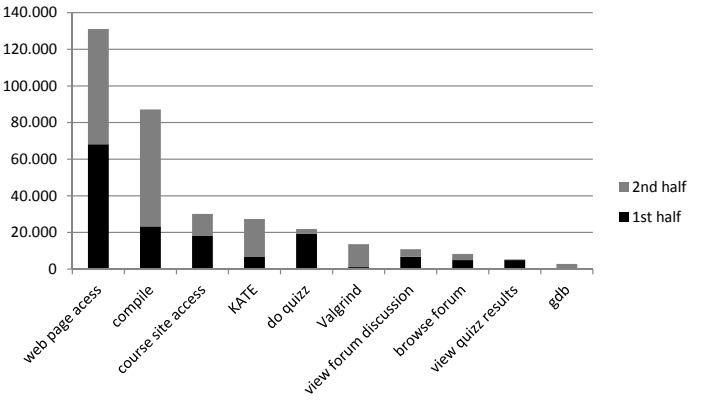

Fig. 2. Ten Most Frequent Events of the Whole Course, for the first and the second half of the semester

Figure 2 shows the distribution of the ten most frequent events. Note that we subsumed some events into one event for this diagram, e.g. *start KATE* and *close KATE* are presented as *KATE*. The top event is *web page access*. This is not surprising in the first and more theoretical half of the course as the students need to access a lot of learning material and need to get familiar with the system and the tools. We could for example observe that the students often need to look up Linux commands. The graph also shows that the number of compile events rose drastically in the second and more practical half of the course.

Figure 3 shows the normalised activity per group for the two parts of the semester and the normalised average group size in comparison. The average amount of events conducted by a single student is 1,019 (470 / 586). Group B-2 is the group with the most active students with on average when looking at the whole year (2063 events per student). This results in an activity score of $2063/1019 = 2.02$. In the first half, group Group A-2 is the most active one with an activity score of 1.55 and in the second half, Group B-2 is the most active one with an activity score of 2.32. The least active group for the whole year as well as the two halves is Group E-2 with an activity score of 0.61 (0.55 / 0.64). This means, wherever the activity score of a course is higher than 1, its students are more active than the average and vice versa. It is noticeable that in many cases the lab groups that belong to the same theoretical group have a similar activity score although the lab groups are not supervised by the same teachers.

Each lab group comprises on average 28.7 students. Group B-2 is not only the most active but also the smallest group with only 10 students and a group size score of $10/28.7 = 0.35$. Group E-1 is the biggest group with 38 students and a

[3] Given the way the data are collected, there is no simple way to distinguish the causes. Clarifying which one would require some extreme, highly unfeasible measures such as taping them in and out of class.

group size score of 1.32. The figure shows that in many cases in which the group size is below the average the activity of the students is above the average and vice versa.

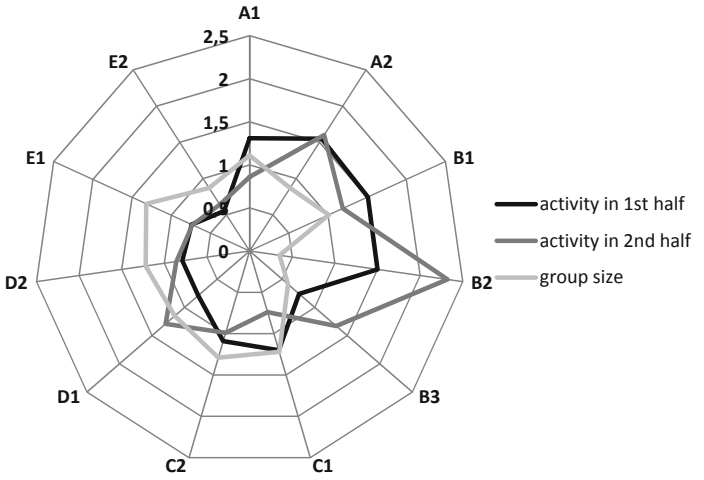

Fig. 3. Web Graph Comparing All Activities per Group to the Group Size

4 Extracting Key Actions

The n-gram approach has been applied to the complete data set as well as to the several sub-sets of theoretical units and lab sessions. When extracting n-grams for a user or a group, we first gather all actions using the userId(s) and order a user's actions depending on the actions' time stamps. Additionaly we take the sessionIds into account to not combine actions conducted in different sessions or by different users.

The base of an activity is usually a complete CAM instance, i.e. time stamp, student ID, and event type (e.g. visitUrl, start, send, etc.) together with an item (e.g. a URL, a terminal command, a file, etc.) and a tool (e.g. editor, browser, etc.). All activities have a time stamp and a student ID and most of these activities have tool, event type and item as well. Some however, such as starting or ending the editor or the debugger, do not have an item.

By specifically taking some aspects out of the calculation, results can be made more general. The less information is taken into account, the more general the results are. In our experiments the calculations were done for several granularity levels: (a) tool, event type and item, (b) tool, event type and item where URLs were shortened to their domain and C files ignored, (c) tool and item, (d) event type only and (e) item only. As explained, shortening the URLs to their domain allows a broader, more general combination of actions. Several students might use Google to find information about the same task, their query term though might differ. The action they execute, however, is essentially the same. This

principle also applies to actions where the names of the C files are taken out as students may use any name for their files, the fact that they compile a C file, however, stays the same. During all calculations key actions of length one and two were discarded as they had not been deemed meaningful enough by the teaching staff of the course.

After extracting the key actions as described above *tf*idf* weights were calculated for the theoretical units as well as the lab groups with the whole course serving as the corpus. Then the 10 most frequent key actions of each calculation were juxtaposed with the 10 highest weighted key actions of the *tf*idf* results. The original key action extraction results and the *tf*idf* results were then given to the course's teachers for a qualitative evaluation.

5 Qualitative Evaluation of the Results

As we claim the hypothesis that key action extraction is a very useful form of data distillation, we gave all analysis results (visualised as in Figure 4 and as text files) to the course's teaching staff and asked for feedback. We especially wanted to know what they can deduce from the results and whether they think they are useful or not and why. The following paragraphs summarise the feedback of the different teachers about their lecture and lab groups as well as the whole course.

In the calculations for the whole course, many extracted key actions dealt with the quizzes. The quizzes are directly embedded in the course notes and are not mandatory to fill and can thus be deployed to measure the level of spontaneous engagement with the course material. Very often students attempted to take a quiz, presumably got a wrong answer and, instead of attempting the quiz again, subsequently clicked on *show result*. For the teachers, this indicates that many students not only did not know the answer of the first two questions of a quiz but also that they needed – or wanted – to "be told" the answer instead of just trying with a second option. Figure 4 shows a visualisation of such extracted key actions.

The key actions reveal those questions that had to be repeated more often, and therefore posed a more complex problem to students. This was deemed very valuable information by the teachers. They concluded the following: The key actions point to those questions that were interacted with the most and thus should be reviewed in class, covered in more detailed or even discussed in the forum. Another aspect of the quizzes is that they show the level of engagement of students with the material. If key actions do not reflect quiz usage, then students are not taking enough quizzes. The high frequency of pointer related questions in the key actions, for example, denotes to the teachers that this is a block that requires special treatment such as reviewing the material in an extra class, complementary course notes, additional exercises, etc. In the key actions, not only the reflection of answering quizzes, but also the rate of failures and redoing of quizzes helped the teachers to understand what they can change in upcoming courses. Also, the differences between doTest-key action-frequencies among the different groups gave the teachers a reference point to see if they should improve student engagement within their groups.

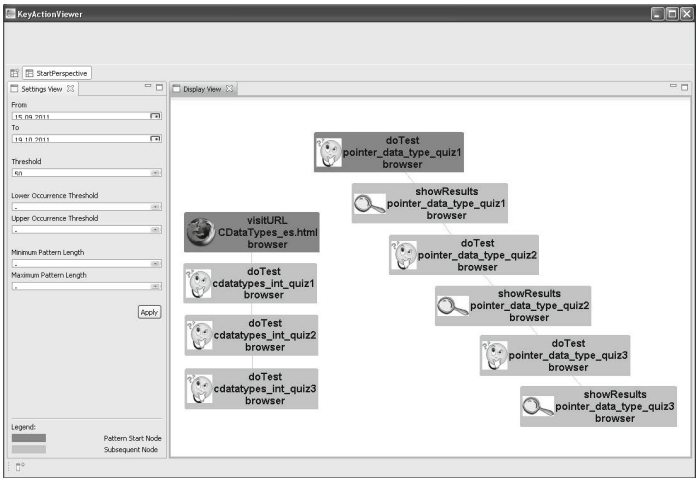

Fig. 4. Visualisation of Some Key Actions from the Whole Course

From the calculations where the C files had not been taken to account, the teachers noticed that there are fairly long and frequent compilation chains. As one of the main objectives of the course is for the students to be able to write applications in C, this was deemed a good result. Another interesting factor was the comparison of the students' compile behaviour from the different theoretical units and lab groups. For three of the five theoretical units the longest key action sequences were compile activities. For the other two the longest sequences came from the browser. The teachers interpreted this as a possible lower engagement in the course for these two units.

Looking at the different lab groups gave them even more detailed insights. For five groups, the longest sequences were compilation activities. These groups were deemed "on track". Another five groups had such activitites as second largest sequences after browser related sequences. That was seen as accaptable. The one group that had the compile activities only in the fourth largest sequences was seen as lacking behind and needed to improve. The teachers agreed that in general they would like to see compile actions as the longest sequences for every lab group. Figure 5 shows the normalised compile activity per group. As can be seen in comparison to Figure 3, the number of compile activities and all activities correlate as do the observations with the longest key action sequences.

The calculated key actions supported what was also reflected in the statistic analysis: the debugger was not used as frequently as it should have been, according to the teachers. For them, without the need of a much deeper analysis, these results reveal that the debugging tool is used significantly below expectations. A straightforward consequence of these results is that changes need to be introduced on the activities to lower the adoption threshold for this tool.

Many lab groups generated most of their key actions related to programming in the second half of the course. This behaviour coincides with the assignment of

the course's main project that is due at the end of the second half. The key actions revealed that a relevant amount of students do not really start working on actual programming tasks until they have the obligation to work on the project. The teachers, however, would have liked to see more programming activity already during the weeks before the assignment.

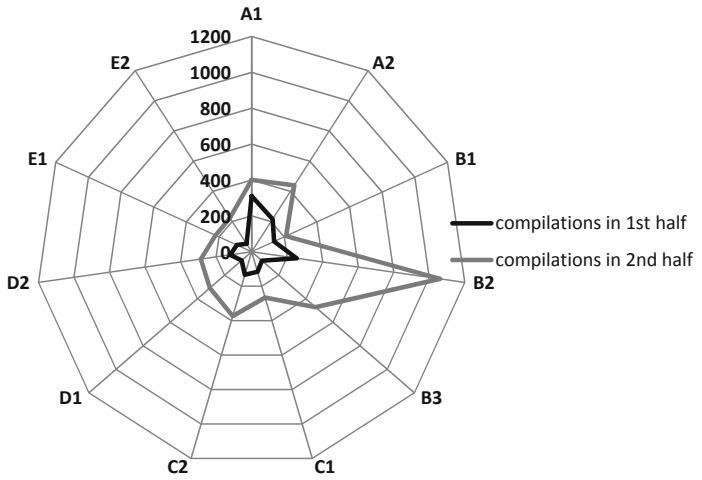

Fig. 5. Web Graph Showing Compile Activities per Group

One teacher noticed that the use of the memory profiler in his lab group had a high *tf*idf* weight as did viewing forum discussions which was a good sign for him. Other sequences that had rather some *tf*idf* weight included accesses to the SVN material of the course. From this the teacher guessed that the students did not put enough effort into learning the command syntax but just copied and pasted it when they needed it. He appreciated this insight planning to work on this topic more in his class.

Looking at *tf*idf* results calculated for the ten students with the highest final mark revealed that they very frequently use both the compiler and the memory profiler. When looking at the results for the ten students with the lowest final mark, such key actions are not weighted high, instead they check the LMS, use the text editor, surf the web and so forth. Teachers noticed a striking absence of the compiler. The *tf*idf* results also showed that the students with higher scores check the content of the forum more often, whereas students with lower scores tend to consult course notes, notices, etc., but not so much the course forum. What seems meaningless but was found remarkable by the teachers was the fact that students with good final marks regularly checked the file containing the course scores (as was reflected in their key actions), whereas students with low final marks barely did so. They deduced that checking for the scores more frequently denotes a higher level of commitment to the course.

In many cases teachers found the information given in the *tf*idf* results more interesting than the plain key action results. They said that computing the key actions alone does give a first insight, but this insight is much more relevant when followed with the *tf*idf* computations because in there, more meaningful sequences appear that can be correlated with student behaviour and also allow for a better comparison of the chosen group to the whole course.

Summing up it can be said that the teachers think key actions to be a very useful form of data distillation. They were able to use the results for course evaluation and liked getting better information from the key actions than from the logs themselves. According to them there were "lots of eye-opening things in there". As the analysis was done in retrospect, the teachers are looking forward to employing the analysis during a course.

6 Conclusion and Future Work

In this paper we presented our approach of key action extraction as a means of data distillation for a collection of contextualised attention metadata. The approach was implemented in a large test bed as part of a C programming course at a university. Student activity data have been recorded, from these key actions have been extracted which in turn have been used by the teachers for the evaluation of the course, that is, for understanding how the students actually dealt with their exercises and how the course can be improved according to their behaviour. As the reactions of the teachers show, the key actions and especially their subsequent *tf*idf* weighting give interesting insights into the course that have not been provided otherwise.

Although we managed to answer the question whether key action extraction results are useful for the teachers at all, new questions arise: Will it be possible to apply the analysis during the course instead of in retrospect? Will this help teachers to react directly and adapt their teaching? Another important question is how to provide the students with their own data. Can it help them to self-reflect on their learning activities? Do they perceive the system's and the teachers' feedback as useful?

In order to answer these new question, the key action extraction will be implemented into the running system of the next fall semester's C programming course. From this we expect to gain even more insight of the usefulness of key action extraction. Apart from providing students and teachers with the results throughout the course and collecting general feedback impressions during several stages, we will also employ questionnaires with which we especially want to address privacy and security issues that are wont to arise.

Acknowledgments. Work partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 231396 (ROLE project), the Learn3 project (TIN2008-05163/TSI), the eMadrid project (S2009/TIC-1650), and the Acción Integrada DE2009-0051.

References

1. Siemens, G., Long, P.: Penetrating the Fog: Analytics in learning and Education. Educause Review 46(5) (September/October2011)
2. Schmitz, H.C., Scheffel, M., Friedrich, M., Jahn, M., Niemann, K., Wolpers, M.: CAMera for PLE. In: Cress, U., Dimitrova, V., Specht, M. (eds.) EC-TEL 2009. LNCS, vol. 5794, pp. 507–520. Springer, Heidelberg (2009)
3. Zinn, C., Scheuer, O.: Getting to Know Your Student in Distance Learning Contexts. In: Nejdl, W., Tochtermann, K. (eds.) EC-TEL 2006. LNCS, vol. 4227, pp. 437–451. Springer, Heidelberg (2006)
4. Kosba, E., Dimitrova, V., Boyle, R.: Using Student and Group Models to Support Teachers in Web-Based Distance Education. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 124–133. Springer, Heidelberg (2005)
5. Mazza, R., Dimitrova, V.: CourseVis: A Graphical Student Monitoring Tool for Supporting Instructors in Web-based Distance Courses. Int. J. Hum.-Comput. Stud. 65(2), 125–139 (2007)
6. Jovanovic, J., Gasevic, D., Brooks, C.A., Devedzic, V., Hatala, M.: LOCO-Analyst: A Tool for Raising Teachers' Awareness in Online Learning Environments. In: Duval, E., Klamma, R., Wolpers, M. (eds.) EC-TEL 2007. LNCS, vol. 4753, pp. 112–126. Springer, Heidelberg (2007)
7. Schmitz, H.C., Kirschenmann, U., Niemann, K., Wolpers, M.: Contextualized Attention Metadata. In: Roda, C. (ed.) Human Attention in Digital Environments, pp. 186–209. Cambridge University Press (2011)
8. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic Keyword Extraction from Individual Documents. In: Berry, M.W., Kogan, J. (eds.) Text Mining, pp. 1–20. John Wiley & Sons, Ltd. (2010)
9. Hulth, A.: Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Stroudsburg, PA, USA. Association for Computational Linguistics, pp. 216–223 (2003)
10. Dörre, J., Gerstl, P., Seiffert, R.: Volltextsuche und Text Mining. In: Carstensen, K.U., Ebert, C., Endriss, E., Jekat, S., Klabunde, R., Langer, H. (eds.) Computerlinguistik und Sprachtechnologie, pp. 425–441. Spektrum, Heidelberg (2001)
11. Scheffel, M., Niemann, K., Pardo, A., Leony, D., Friedrich, M., Schmidt, K., Wolpers, M., Delgado Kloos, C.: Usage Pattern Recognition in Student Activities. In: Delgado Kloos, C., Gillet, D., Crespo García, R.M., Wild, F., Wolpers, M. (eds.) EC-TEL 2011. LNCS, vol. 6964, pp. 341–355. Springer, Heidelberg (2011)
12. Niemann, K., Schmitz, H.C., Scheffel, M., Wolpers, M.: Usage Contexts for Object Similarity: Exploratory Investigations. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, LAK 2011, pp. 81–85. ACM, New York (2011)
13. Romero Zaldívar, V.A., Pardo, A., Burgos, D., Delgado Kloos, C.: Monitoring Student Progress Using Virtual Appliances: A Case Study. Computers & Education 58(4), 1058–1067 (2012)