

Chapter 2

An Approach to Measure Coding Competency Evolution

Toward Learning Analytics

Vive Kumar, Kinshuk, Thamaraiselvi Somasundaram, Steve Harris, David Boulanger, Jeremie Seanosky, Geetha Paulmani and Karthikeyan Panneerselvam

Abstract There is a great deal of interest in the area of learning analytics and their use in assessing both student and content success in a digital learning environment. This chapter describes the results of a pilot study that used a combination of software tools and processes to collect the data generated from students taking an introductory C programming course, and their interactions related to specific study activities. This study helps determine whether it is possible to collect enough useful analytics to begin to identify constructive learning practices and strategies that determine competency evolution. We look at the types of learning traces collected through our system and then discuss how the data might be used to provide insight into student, class, or content actions. We then consider how additional analytics may be collected and used to supplement our initial results. The technologies used in this study address requirements of big data learning analytics where the data come from (a) the learner and his/her immediate surroundings; (b) the social network of the learner related to the learning tasks; and (c) the environment that inherently encases learning activities. While the data are used to measure competencies and their evolution over a period of time, the resultant profiles show the evolutionary processes students have adopted in developing specific competencies.

Keywords Learning analytics · Big data · Coding · Novice programmer

V. Kumar (✉) · Kinshuk · S. Harris · D. Boulanger · J. Seanosky · G. Paulmani
School of Computing and Information Systems, Athabasca University, Athabasca, Canada
e-mail: vive@athabascau.ca

T. Somasundaram · K. Panneerselvam
Madras Institute of Technology, Anna University, Chennai, Tamil Nadu, India

2.1 Introduction

Students consume an array of media in their studies, whether in classroom or in online instruction. The media includes slide shows, reading material, audio/video lectures, interactive tutors, and automated tests. As they work through the course material, they generate digital learning activity traces that track their interaction with the course material, much like a Web analytics system tracks visitors to a Web site. Similarly, as students complete the associated computer programming assignments, software development traces are generated which can be used to track the student success and coding competency.

Software development traces generated by students as they solve computer programming problems, such as number of compiles, warnings, and various errors, can be captured from their development environments, along with the student learning traces from the digital course materials accessed for the duration of the course.

In this pilot study, a software analytics system called HackyStat¹—an open source project originally developed by researchers in the University of Hawaii—collects software development activities and logs from a variety of development environments, including Eclipse, EMACS, and Visual Studio. In addition, an open source Moodle plug-in developed by the University of Las Palmas de Gran Canaria called Virtual Programming Lab (VPL)² was altered to perform a similar data collection task from within Moodle itself, for less intensive programming exercises. Moodle's reporting module itself offers data traces on student interactions with the learning material housed in that system as well as performance measures of learners.

Using a combination of static analysis tools and toolkits, depending on the language in use (e.g., Java program analysis uses a combination of Checkstyle, FindBugs, and JUnit, while C programs use a combination of Lint and Virtual Programming Lab (VPL) customizations), coding activities of students can be logged at finer levels of granularity. Coding activities include code design, writing of code, efforts of debugging, attempts at documenting the code, and testing of the code. Using both manual and automatic grading, the work of the students can also be graded at finer levels of granularity. All these datasets can be captured in HackyStat.

The HackyStat system allows for the greatest flexibility in data collection and retains the raw data in a SQL database for further processing. The collected data include not only the final product, a software program, but also the intermediate steps, such as designing, coding, debugging, testing, and documenting, that the student had taken to reach the final product. Also, because of its open source nature, additional HackyStat plug-ins can be developed or altered to track programming in other environments or languages, as required.

¹ <http://code.google.com/p/hackystat/>.

² <http://vpl.dis.ulpgc.es/>.

The raw datasets are used to populate a coding competency ontology (based on Java and C ontologies³ from the University of Pittsburgh). The ontology contains two parts: first, the establishment of a hierarchical framework for the domain knowledge that primarily includes classes of programming actions, traces from the learning materials, types of programming problems, and primary relations between these categories and, second, a causal mechanism that attempts to develop models of causation in coding-related activities.

2.2 Research Rationale

The goal of this research is to make inferences about student's coding competency that may establish causal relationships between student activities and learning outcomes and to identify and share inferences that seem to result in constructive coding practices for students. Additionally, we are also interested in the identification of subgroups of students for whom different learning strategies lead to optimum coding competency or that certain types of materials are more effective learning tools than others. Further, suggestions could be proposed to the student, perhaps through the use of a smart agent, to focus on certain materials and strategies based on their performance and that of past students. We reviewed literature to have an understanding of the technological and conceptual hurdles that need to be tackled in achieving these goals. The following summarizes our review efforts.

Whether observed or not, data are being continually produced in all phases of learning whether it is classroom-based lecture or an online discussion or a blended training exercise or lifelong work scenarios. Instructional design efforts generate their own sets of data concerning course design models, development workflow, course guidance, adaptation opportunities, course comprehensiveness, and course quality. Irrespective of where, why, how, and when learning happens, learning environments do create new learning experiences and correspondingly new sets of data.

Each learning experience can be measured as long as the underlying data are collected. Be it personalized learning, formal learning, ad hoc learning, classroom learning, or mobile learning, different types of learning environments offer different types of learning and instructional attainment datasets. The associations that evolve among these datasets on a day-to-day basis have conveniently and predominantly been ignored since the beginning of modern era. For instance, from a learner's perspective, summative assessments, augmented with formative assessments, are used to only estimate competency of the learner. The capacity of the learner and the effectiveness of learning processes undertaken by the learner are not normally measured. This is not because of the challenges involved in obtaining and validating data on capacity and effectiveness, but because of the pervasive nature of 'lethargy' in teaching.

³ <http://www.sis.pitt.edu/~paws/ont/java.owl> and http://www.sis.pitt.edu/~paws/ont/c_programming.rdfs.

Learner competence can be measured to a greater extent by using a variety of assessments. Learning capacity, on the other hand, depends on a number of personal factors including working memory, motivation, support, learning material, and learning styles. Learning effectiveness is a measure that mainly involves competence and capacity. Capacity and effectiveness have generally been ignored by contemporary instruction attributing to the lethargy mentioned above. Expectation of the 'bell curve' as the norm of classroom performance is another factor that attributes to lethargy in teaching. Constraints placed on learning outcomes force instructional processes to be satisfied mostly with summative feedback-based associations, offering further attributions to lethargy.

Learning processes/activities undertaken by individual learners from the time a concept to be learned is introduced, up to the time instructional efforts cease to teach that concept to that learner can potentially be observed using a variety of techniques and technologies. For instance, the behavior of classroom interactions of students attending a lecture can be video-recorded and analyzed at real time to estimate the level of comprehension of a set of concepts introduced in the lecture. The behavior of online interaction of students in a discussion forum can be analyzed at real time to estimate their writing competence. The behavior of social interactions of students can also be analyzed to estimate their sentiments on specific services offered by the institution related to learning. These exclusive observations, along with other traditional data, can supply the 'raw data' for learning analytics.

Raw data are being produced at an alarming rate in other domains including transportation, health care, geographic advertising, retail, banking, and energy (Manyika et al. 2011). A report on online learning in Canada⁴ outlines the market share of learning management systems (LMS) that already produces large quantities of data on learning and the inclusion of online learning as part of core business plan of academic institutions across Canada.

The following sections review techniques, technologies, and systems that specifically address raw data collection with a view to big data analytics.

2.2.1 Techniques, Technologies, and Systems

An infrastructure (Bienkowski et al. 2012) that supports learning resource discovery, sharing, and amplification indicates to a large volume of metadata and data that are not usually barricaded from public view. At-risk students can be recognized based on data about the type of risk, interventions based on predictive models can be designed to mitigate that risk, and the utility of the models can be gauged by the trace data on applied intervention (Essa and Ayad 2012). A number of data-driven analytics drivers to mine effectiveness of learning have been reported for use in learning and academic analytics (Ferguson 2012).

⁴ <http://www.contactnorth.ca/online-learning-canada>.

Social learning analytics offers access to a slew of data on informal conversations among students and instructors (Shum and Ferguson 2012).

Raw data are not confined only to formal and informal datasets, but also include ad hoc datasets with marginal overlap with learning. Ad hoc datasets include browsing patterns, reading habits, writing style (including freehand writing), coding, posture analyses, collaboration drivers, thinking protocols, chats, domain-specific tool traces, recording of cognitive and metacognitive traces, and knowledge traces.

Among many others, raw data can be processed toward data mining (Romero et al. 2008), program evaluation (Hung et al. 2012), real-time classroom feedback (Wood et al. 2013), prediction (Watson et al. 2013; Abdous et al. 2013), visualization (Mazza and Dimitrova 2003; Kim and Lee 2013), and modeling (Medeiros et al. 2013), among others.

Potential observations and analyses of the aforementioned categories would shed light on instructional processes that indicate introduction, deliberation, assimilation, evaluation, and application of a concept or skill to various degrees by different types of learners. However, a holistic view of the rates of changes in adaptations, repetitions, and refinements of instructional processes and instructional resources, as well as the rates of changes in learning processes and learner capabilities, has never been observed as a matter of fact—till now, till the arrival of learning analytics as the science of analysis that concerns learning, and the suite of tools that observe, validate, and semi-automatically associate datasets related to learning and instruction.

Presently, it is not possible to technically trace the evolution of individual skills of programmers through the accumulated study experiences of individual or groups of students, thus necessitating the need for a more effective capture of study experiences of learners. In particular, we plan to use ontologies (Dicheva et al. 2009) since ontologies facilitate the integration of various sources of information covering the same domain knowledge. In our case, we use ontologies to interrelate information about learning objects, learning activities, and learners, captured from various tools embedded within Moodle.

While these technologies focussed mostly about processing readily available data about study experiences of learners, quite a significantly large volume of data about formative study experiences of learners, particularly online learners, goes unnoticed, unutilized, and unappreciated. A number of domain-specific tools target and enhance activities associated with learning (Zinn and Scheuer 2006; Mazza and Milani 2005; Mazza and Dimitrova 2004; Kosba et al. 2005; Brooks et al. 2004; Jovanovic et al. 2009; Shakya 2005; Baghaei et al. 2007; Ghidini et al. 2007).

Competency is an elusive term. There is plenty of research on how to computationally capture it, process it, and foster it (Butler et al. 2008; Boekaerts and Corno 2005; Cross and Angelo 1988; Angelo and Cross 1993). Yet, there is a considerable gap in how computational models capture programming competencies. The aim of this research is to tackle this and develop a widely acceptable computational model for Java competency that can be customized by the individual student.

Study of programming skills is not new. The most traditional method of evaluating student's competency is that of standard examination by way of assignments, projects, laboratories, examinations, and tests. In 2004, researchers Rountree, Rountree, Robins and Hannah (Rountree et al. 2004) utilized this standard method of evaluation to investigate the comparison of difficulty between code generation and code comprehension for introductory Java programming course students. They soon found that this method of evaluation showed 'Contrary to their initial expectations, the code comprehension and generation skills appear to be tracking each other', as well as 'students receive a grade for their work, but this result tells us little about the nature of their knowledge'.

Rao et al. track the growth of competencies by observing how students develop multiple computer programs, augmented with summative as well as formative feedback (Rao et al. 2007). Mixed-initiative interaction (Allen et al. 1999), employed by Rao et al., presents researchers with opportunities to intervene and prompt learners based on certain criteria, where the intervention is initiated by the system. Java learning environments such as MICE and CORE bring into view excellent mediums by which students' development activities can be tracked (Rao et al. 2007).

Novice java programming students write numerous blocks of code and submit them to the compiler for verification where in return the compiler provides them with a summative feedback on the correctness of their code based on a set of accepted complex syntactical style guides (Rao et al. 2007). These articles and their conclusions showcase that programming competency could be tracked and prompted to ensure that students are aware of their current competency levels, aware of what they need to do to improve the levels, and aware of what tools and skills they need to engage to target such an achievement.

As education moves out of the classroom and enters the era of online and distance education, the type of personalized and adaptive scaffolding needed to support different needs of a variety of students is an area of research that is larger than life. This research will contribute to the study of personalized interactions, customizable interfaces, and adaptive instructions as key premises. In summary, this research will study how continuous data collection and modeling of the collected data in online activities related to coding can benefit task-specific competencies for novice programming students.

2.3 What Is Big Data Learning Analytics

Learning analytics is the science of analysis, discovery, and utilization of learning traces in emergent and related levels of granularity. Analysis could include techniques ranging from data mining to machine learning and to big data analysis. The discovery of new relations and the discovery of even new data include unconventional data, for example, the family income of a politically competing region, inherent economic drivers influenced by a curriculum, and rate of changes in motivation levels of students with respect to weather. Relations of interest include

sentiments among learners across collaborating groups, interinstitutional credit transfer policies among institutions, and mutual respect among instructors. Trace data refer to observable raw data of study activities such as reading, writing, conceptualizing, critically thinking, solving problems, storytelling, and visualizing, where a network of study activities leads to a measurable chunk of learning. For instance, the types of sentence openers used by a learner, the range of errors that the student can confidently correct, the level of trust exhibited by the student in sharing information in a forum, and the depth of understanding in a set of concepts are examples of learning traces where one could measure learning over time. In learning analytics, data are expected to arrive continuously, potentially in an interleaved fashion, subject to interpretation at various levels of granularity.

In general, learning traces translate raw data into incoming data for learning analytics, where incoming data are typically big, unstructured, unrelated, and fit multiple models and possibly multiple theories. Importantly, learning traces capture highly personalized study experiences. For instance, consider the example of studying about the notion of a pointed object penetrating better than a blunt object. The study goal is to understand why this is so. A visual-oriented learner may choose to study and explain this phenomenon using visual tools, while a fellow student may work out the mathematics behind pressure and explain the results in terms of equations. Thus, learning traces capture different types of skills and background knowledge exhibited by individual learners, the extent to which learners learned new concepts while studying a particular material, the amount of time (efficiency) learners took to study the material, the effectiveness of assessments to conclude that the concept had been learned efficiently, evidences of learning activities/experiences (e.g., a video of the learner trying a sharp pencil on someone's palm and the learner's answer to the question on pressure in the examination), and resources used by learner (e.g., the pencil, simulation run by the learner). While each learning trace is measurable, there is no standard scale of measurement that applies across different types of traces. A learning trace is the least common denominator for a measure of learning.

Data mining offers techniques that typically operate on well-defined, medium-sized datasets (up to gigabytes). Intelligent tutoring systems typically operate on limited-sized data (up to megabytes). Learning analytics aims to operate on large volumes of data (at least in the order of terabytes, hence the name, big data) as well as large volumes of models produced from the data. The models are dynamic, if not emergent, because more often than not, processing of data would include newly arriving, marginally related datasets. Examples of emergent models include 'reading habits across domains', 'writing styles across contexts', 'application of problem-solving skills across tools' (SPSS, R, Eclipse, MATLAB, and so on), 'social distances while collaborating with peers', 'comfort zone when interacting with instructors', and 'research potential'.

Data can be gleaned with sensors. Contemporary technologies only offer sensors that observe a tiny fraction of data associated with these new learning experiences. In learning analytics, one could conceive of three types of sensors—instructional sensors, context sensors, and internal sensors. Instructional sensors have explicit association with the study as seen in LMSs. Intelligent tutoring offers a wide variety

of instructional sensors such as sensors to observe self-regulation activities of learners and sensors that observe the interactions between learners and anthropomorphic pedagogical agents. Context sensors include the observable environment in which learning takes place. Examples of context sensors include autonomous software agents that act independently on students' behalf and data traces that cluster students from across multiple institutions, based on specific competencies. Internal sensors include methods that estimate learners' ability to think laterally and critically, ability to strategize in a game environment or regulation exercise, ability to use working memory and comprehension techniques, and the ability to pace oneself and proactively study. The observations of these three types of sensors and the interrelations among individual data streams produced by each sensor yield the raw data.

While big data have been a recognized field of exploration in many domains such as traffic regulation, health care, geographic advertising, retail, banking, public sector, consultancy, energy efficiency, and policing, applications of big data techniques in learning are still in their infancy. Learning attainment and instructional attainment need to synchronize with each other to offer an optimal approach to learning. In addressing this very goal, this chapter highlights the application of learning analytics with a case study. The case illustrates the collection of raw data, the transforming the raw data into learning traces, the application of learning analytics techniques on learning traces, the use of results of analytics toward balancing learning and instructional attainment, and the measurement of impact of analytics.

Technology-enhanced learning, in general, implies that learners study learning content and collaborate with their peers and instructors through the use of learning management systems. As part of their study, learners need to solve problems by using development environments built for specific programming languages they study. Learners also often need to use resources publicly available on the Web (not necessarily included into the official course materials) and ask for the help of their peers. However, with the current technology support, students are not able to seamlessly integrate their learning activities when they use different tools. Further, the current technology does not allow formal integration of interaction data collected from across these tools. Similarly, for their instructors, it is very hard to relate the underlying activities their students were engaged in that produced the submitted learning products. This is not conducive for instructors to be able to help students with personalized and context-specific advice. In general, most instructors only receive the final product of students' study efforts; neither the study process students went through nor the intermediate study products and study issues faced by students come to the attention of instructors.

2.4 Datasets and Analysis

We intend first to describe the overall data flow and then discuss about the various technologies employed in the processing of the data all along the way that is the chain of I/O. We first note that a learning analytics system not only

collects data to support the mission of the owner organization but also collects metadata about the interactions that the various stakeholders have with any component of the system. Although the learning analytics framework has been designed to be as generic as possible, we discuss its use in a programming context which makes up a perfect, complex experimental environment to show its potential.

A number of HackyStat sensors have been developed that include Eclipse Integrated Development Environment (IDE) sensor called EIDEE, a Java tutor sensor called MILA, and Moodle-embedded IDE sensor called VPL. In addition, Moodle itself minimally tracks student interactions with course contents through page browsing activities, assessment activities (e.g., quiz), and communication activities (e.g., forum, messages). Figure 2.1 presents an architecture of our approach.

To collect the data in a central repository, HackyStat sensors are embedded within Eclipse, the MILA tutors, VPL, and Moodle. All data are sent to the HackyStat sensor base. The sensor base contains all the data collected from HackyStat sensors. The sensor base is essentially a Derby database.

To extract meaning from the data stored in the sensor base, we build a set of ontologies for each student using the incoming data. The incoming data flow continuously, and hence, the ontology is instantiated continuously as and when the data arrive. For instance, given a specific programming problem, the abstract syntax tree of the student source code will be time-stamped and stored in ontology. This ontology records the state of the abstract syntax tree of the student and the changes it undergoes continually. A second ontology captures errors made by the student during the development of the code. This ontology will store the results generated by code analysis tools such as Eclipse

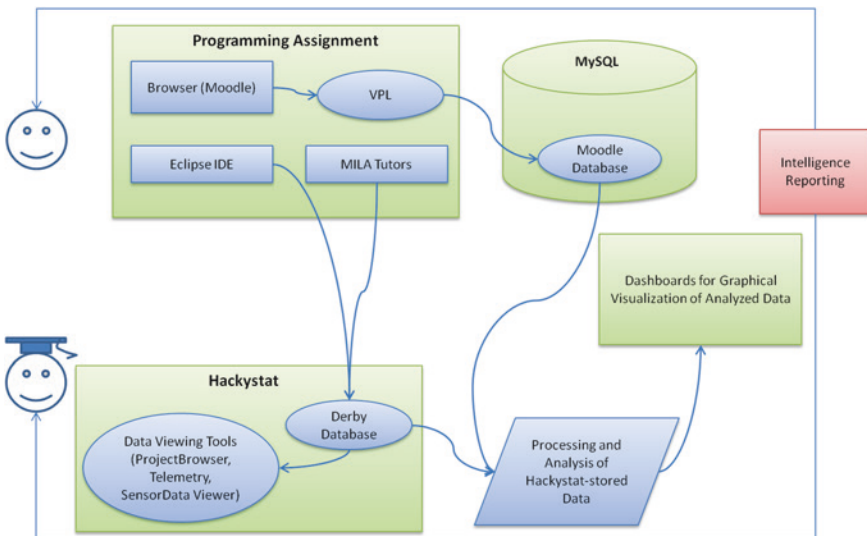


Fig. 2.1 System architecture

JDT⁵/CDT⁶ compilers, FindBugs, CppCheck, and Checkstyle. To handle ontologies programmatically, we use the Apache Jena⁷ framework. Ontologies will be written in the RDF or OWL format depending on the needs of expressivity or power of inference engines. We use BaseVISor,⁸ an RDF-based forward-chaining inference engine, to extract useful inferences from these two ontologies. The outcome of inferences and other statistics obtained from the data stored are presented in a dashboard called MI-DASH.

2.4.1 HackyStat Sensors

HackyStat provides a range of sensors to collect data from tools such as Eclipse, Ant, Checkstyle, Clover, FindBugs, JUnit, Visual Studio, and tools generating XML data. HackyStat also provides a sensor shell Java API which supports the creation of customized sensors to increase the range of tools monitored by HackyStat sensors. The main advantage of the sensor shell is that it has a built-in data integrity mechanism; that is, the sensor sends packets of data to HackyStat at specified time intervals or when an event occurs (clicking a button, pressing the return key, etc.) via a network (be it intranet or Internet). However, if for some reason network connectivity is lost while data are being sent, the sensor shell will temporarily store the data in XML files on the user's local machine until connectivity is restored. Then, any local sensor data are automatically pushed on the HackyStat sensor base. This ensures that no data are lost due to some unexpected loss of connectivity.

Since HackyStat is open source, we modified its Eclipse sensor to suit the particular needs of our experiments. Essentially, we kept the data types originally designed with the sensor and added new ones. The sensor was also modified to fit various versions of Eclipse for various packages (Helios, Galileo, Indigo).

Finally, we created our own Eclipse package with our customized HackyStat Eclipse sensor embedded. That package also contains course assignment projects ready to be monitored by the sensor. The purpose of this package is to hide the complexity of the sensor configurations from the students.

HackyStat is designed to collect data in the background without intervening with the application. For example, once embedded in Eclipse, students would not even notice that HackyStat is collecting data. Alternatively, HackyStat can be programmed to show exactly what data are being collected, when, and how it reaches the Web server for storage. Shown in Fig. 2.2 is HackyStat's project browser that shows data being collected corresponding to a project.

⁵ <http://help.eclipse.org/helios/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Foverview-summary.html>.

⁶ <http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.cdt.doc.isv%2Freference%2Fapi%2Foverview-summary.html>.

⁷ <http://jena.apache.org/>.

⁸ <http://www.vistology.com/basevisor/basevisor.html>.

Timestamp	Owner	SensorDataType	Runtime	Tool	Resource	Type	Subtype	Description	Username
2013-12-18 1:49:11 PM	hackystat@athabasca.ca	DevEvent	2013-12-18 1:49:11 PM	Emacs	BeijingNormalUniversity/CMP277/Assignment2/Car.java	Edit	StateChange	Generated when the active buffer has changed.	Guang Chang
2013-04-16 2:41:14 AM	hackystat@athabasca.ca	DevEvent	2013-04-16 2:41:14 AM	Emacs	ChileUniversity/NF3073/Project/Hospital.java	Edit	OpenFile	Generated when a new file is opened.	Gustavo Baloian
2013-05-09 4:06:44 PM	hackystat@athabasca.ca	DevEvent	2013-05-09 4:06:44 PM	Emacs	AnnaUniversity/MIT-C/Assignment/PetrolPump.c	Edit	SaveFile	Generated when the current file is saved.	Konijeti Rosaiah
2013-10-01 2:32:33 AM	hackystat@athabasca.ca	DevEvent	2013-10-01 2:32:33 AM	Emacs	TaiwanUniversity/Java/Assignment2/Robot.java	Edit	CloseFile	Generated when the current file is closed.	Pan-Chyr Yang
2013-10-06 5:16:32 AM	hackystat@athabasca.ca	DevEvent	2013-10-06 5:16:32 AM	Emacs	AthabascaUniversity/COMP268/TMA1/Calculator.java	Edit	LExit	Generated when Emacs is exited	John Smith
2013-02-19 12:09:44 AM	hackystat@athabasca.ca	DevEvent	2013-02-19 12:09:44 AM	Eclipse	BeijingNormalUniversity/CMP277/Assignment2/Car.java	Edit	StateChange	Generated when the active buffer has changed.	Guang Chang
2013-02-19 11:24:04 AM	hackystat@athabasca.ca	DevEvent	2013-02-19 11:24:04 AM	Eclipse	ChileUniversity/NF3073/Project/Hospital.java	Edit	OpenFile	Generated when a new file is opened.	Gustavo Baloian
2013-07-25 10:37:19 PM	hackystat@athabasca.ca	DevEvent	2013-07-25 10:37:19 PM	Eclipse	AnnaUniversity/MIT-C/Assignment/PetrolPump.c	Edit	SaveFile	Generated when the current file is saved.	Konijeti Rosaiah
2013-03-16 6:48:51 PM	hackystat@athabasca.ca	DevEvent	2013-03-16 6:48:51 PM	Eclipse	TaiwanUniversity/Java/Assignment2/Robot.java	Edit	CloseFile	Generated when the current file is closed.	Pan-Chyr Yang
2013-06-13 4:59:43 PM	hackystat@athabasca.ca	DevEvent	2013-06-13 4:59:43 PM	Eclipse	AthabascaUniversity/COMP268/TMA1/Calculator.java	Edit	Exit	Generated when Eclipse is exited.	John Smith
2013-02-20 1:08:29 PM	hackystat@athabasca.ca	DevEvent	2013-02-20 1:08:29 PM	Eclipse	BeijingNormalUniversity/CMP277/Assignment2/Car.java	Edit	ProgramUnitAdded	Generated when adding a new program construct such as an import statement, class, field, or method.	Guang Chang
2013-10-13 9:48:16 PM	hackystat@athabasca.ca	DevEvent	2013-10-13 9:48:16 PM	Eclipse	ChileUniversity/NF3073/Project/Hospital.java	Edit	ProgramUnitRenamed	Generated when renaming a program construct.	Gustavo Baloian
2013-01-01 8:09:53 PM	hackystat@athabasca.ca	DevEvent	2013-01-01 8:09:53 PM	Eclipse	AnnaUniversity/MIT-C/Assignment/PetrolPump.c	Edit	ProgramUnitRemoved	Generated when removing a program construct.	Konijeti Rosaiah

Fig. 2.2 HackyStat collects data and records it for web browsing

The Eclipse sensor enables to collect different data types during code development. The three types of data captured by the sensor are Edit, Build, and Debug. It tracks any operations made on a file such as opening, saving, or closing a file. It also records any addition, renaming, removal, and move of programming constructs. Any failed compilation is also sensed and sent to the sensor base. The sensor also reports debugging activities such as the start and termination of debugging sessions, the setting and removal of break points, and the passage of the debugger into or over a block of code. We also modified the sensor so that the source code may also be collected as the student types. The sensor collects the source code at 1-s intervals or whenever the student compiles his/her code (saves the code file) so that we might reconstruct his/her code at any time.

As for the MILA tutor, we decided to build a sensor from scratch using the sensor shell API. The MILA tutor records the student source code as he/she presses the up/down arrow keys or the return key. There are different tutors for different programming problems. Each tutor consists of a client-side Java Web Start application and a server-side Java servlet. The Java Web Start application stores data collected from the student in a buffer and sends the data over HTTP post messages to the Java servlet. The servlet contains a customized sensor which sends to the HackyStat sensor base all the instances of data types collected by the client application. The servlet also parses the final version of the student's source code to generate the corresponding abstract syntax tree and compiles the code using the

Eclipse JDT/CDT compiler. The tutors capture every source code change, build the code, and parse it to track the evolution of code and errors during problem solving. Code change data and error data are sent to the HackyStat sensor base. Source code is captured as a DevEvent sensor data type, while errors are captured as CodeIssue sensor data types.

As for the Moodle Web sensors, we note that the HackyStat sensor shell (as well as all of HackyStat) is built in Java, but since Moodle is built using PHP, JavaScript, and AJAX, we use a framework called PHP/Java Bridge which is designed to allow the integration of Java libraries inside PHP code and vice versa. Therefore, we use the sensor shell Java API inside the Moodle PHP code to build a HackyStat sensor for Moodle that uses the data integrity feature of the Java sensor shell.

We create event handlers in Moodle’s code that incorporate the Web HackyStat sensors which send data about the event that just occurred (action type, time stamp, action context, any user input, etc.) to the HackyStat sensor base.

2.4.2 Data Analysis

Discussion around the analysis of the data is centered on the creation of key performance indicators of learning analytics that fall along three distinct dimensions: overall course measurement, student-based traces, and content-specific traces.

With the ability to collect formative data on student learning activities, data on challenges faced by students, data on ways in which students reacted to these challenges, data on the quality of instructional supported afforded to students, and data on the means through which effectiveness of instruction is measured could be traced and collected. These traces offer valuable information about learner capabilities, instructional effectiveness, and the overall quality of instructional environment. For example, a student profile on coding skills can be built as shown in Fig. 2.3.

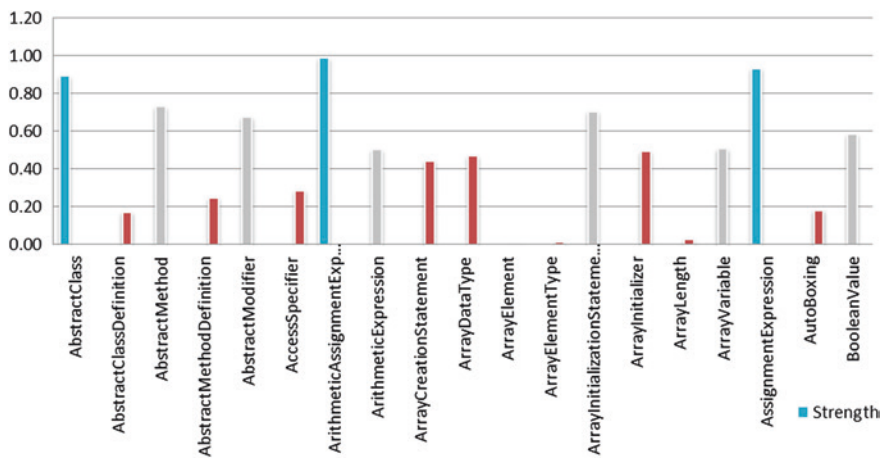


Fig. 2.3 A student profile on coding competency

What they do not offer are the means through which the instructional gaps could be addressed. For instance, if a student struggled to develop appropriate test cases, these traces provide clues on these struggles. However, each student (or a group of students with similar issues) needs individualized instruction that adapts to his/her specific requirements. This is an opportunity for interaction with the student to address specific instructional gaps. With this in mind, we developed additional options within each tool that students could use to engage in these interaction opportunities. For instance, within each MILA tutor, we built opportunities for a type of self-regulation and co-regulation. Similarly, the Eclipse IDE itself is extended with query interfaces for students to supply data on their knowledge gaps and seek information about others who might have faced similar gaps and how they proceeded to address them. This is possible because HackyStat and its sensors continuously collate information on progress of students in specific coding activities and groups them using a particle swarm optimization algorithm.

Here, we provide some sample outcomes of the system. First, one could get simple statistics such as the number of learner interactions within Moodle with respect to the course. An interaction is the activation of any study tool within Moodle. We observed that over a period of a semester, 182 students out of the total 240 recorded 42,781 interactions within Moodle. During the same time period, these 182 students also attempted 11,151 times to write code using VPL. Eventually, these 182 students submitted 1,147 complete programs for evaluation. These 182 students also attempted to take self-study quizzes 1,693 times and viewed course content 7,631 times.

We can then dig deeper to extract study sequences of successful students. For instance, based on the marks received in the first assignment, one could gather the number of times successful students (say, those who scored 90 % or above in the assignment) attempted to solve each programming problem. We can then allow a student to compare his/her attempts with that of the successful student. That is, the student can perform a line-by-line comparison of code between each of his/her attempt to write the program with that of a successful student.

The student can also compare the time he/she took to solve a problem versus the average time of a successful student. Digging deeper, the student could investigate the exact situation that caused the difference in problem-solving time. Thus, the system provides query interfaces for participants to extract key information and then allows the participants to engage in guided, mixed-initiative reflective interactions for a deeper understanding of issues.

Further, the system allows participants to engage in regulatory activities where study strategies and instructional strategies can be recorded and followed. For example, a student participant can look at the average number of significant errors that he/she created. Based on this, he/she can investigate how long he/she took to resolve each of these errors in comparison with other standard metrics. The student can then set himself/herself a goal that allows him/her to solve errors faster. The system will continue to monitor progress made by the student and offer him/her proactive feedback on the achievement of this goal.

The sequences through which students attempt to negotiate learning can be captured using a variety of sensors. The captured data allow individual students or instructors or even the organization itself to supplement students' study activities with additional instructional tools. The learning analytics system, by itself, does not prescribe to any set of tools but allows users to include (or exclude) any tool of their choice. Each included tool will be required to collect formative data as per the specifications. Thus, our vision is not to prescribe any suite of tools but to encourage users to pick and choose their own choice of tools, as long as each tool contributes to data capture through HackyStat.

The choice of tool customization could be based on the needs. For example, with respect to content quality, one could ask—'Is there a reduction in coding errors by a student who first watches the lesson and then attempts the program versus a student who simply attempts to answer the problem (so, does watching the lesson improve the student's programming)?' Other sample questions are—'Is there any difference between a student that views the videos, attempts the question, and then does the programming exercise, versus a student who does not?' and 'Is there an improvement in coding submissions over time, as the student becomes more proficient?' As mentioned before, such questions can be posed by any participant (e.g., a student or an instructor) and answers can be received from the system in real time. A participant may choose to investigate further to explore the types of students who benefit from a particular sequence of study activities, to gain more instructional insights. Thus, the range of application of analytics goes from looking specifically into one particular competency development for one student to a trend in learning from large groups of learners.

2.5 Results and Contributions

One of the main benefits of the HackyStat/VPL/Eclipse/Moodle customizations is the ability to collect and store the raw data for later processing—and even to combine datasets over time, where appropriate, to gradually increase our total sample size over time. Thus, we are able to collect an increasing amount from a number of different classes, with almost no intrusion into the learning environment from the perspective of either the student or the instructor.

Further, as both VPL and HackyStat are multilingual and open source, we are able to deal with data from a number of different programming languages, including C, C++, Java, and Python, and even have the ability, with some additional customizations, to collect data from students studying across different languages.

The main anticipated technological contributions are twofold—one, to provide an ontology-based data mining/analytics framework, and two, a causal model for developing inferences from the relationship between learning activities and coding competency. These inferences may involve optimal learning strategies, an optimal mix of learning media, or may identify subgroups with different learning characteristics and requirements.

A secondary but equally compelling contribution is the development of a comprehensive software traces platform reporting on the effectiveness of digital introductory computer programming course material and study habits and feeding that data back to both students and instructors.

Through the combination of these two results, a third goal is the creation of an intelligent agent that can monitor the data sources on a student-by-student basis, to identify successful patterns of learning, based on a student's interaction with the digital learning environment, and can suggest corrective actions, materials, and behaviors that might improve performance, based on the past results of similar student behaviors.

Learning analytics is not about guiding students in a particular, well defined study pathway. Instead, it is about guiding students to explore meaningful and personalized study pathways that have a higher potential of success.

The data streams used in learning analytics are neither finite nor well defined. Instead, learning analytics allows any tool to offer a new data stream that can be of use. API specifications are being developed to ensure data compatibility, semantic coherence, and tool integration.

Learning analytics does not aim to offer solutions to problems encountered by students as they study. Instead, it offers students to find study pathways that might lead to solutions.

Learning analytics is mostly about the 'ING' in learnING—the process of learning and how it arrives at products and outcomes. For example, along with the review of the thesis draft submitted by a student, a supervisor could analyze the processes/pathways the student took to reach that draft. This analysis includes the articles the student referenced, passages he/she had quoted/used in the draft, passages written by the student, ideas independently generated by the student, whether references resulted in a modification of certain ideas presented in the draft, the pace of writing over the entire duration, the impact of the supervisor's discussions with the student during the development of this essay, and so on. Thus, learning analytics aims to offer as well as discover useful clues that help students aim at better learning capacity and instructors aim at effective instruction.

The technologies we have currently included range from simple statistics, to rule-based approaches to pattern recognition, to mixed-initiative conversational agents, and to causal models that discover causal connections between study pathways and competencies. The technologies are embedded in a learning analytics platform that is both extensible and open. Anyone could add a new piece of tool/technology that offers useful data to the learning analytics platform. Further, any tool/technology that has been submitted is also made available for others to use.

We are at the cusp of a significant change in perspective on learning that not only offers to guide students to specific solutions but also allows them to reflect/regulate their solution pathways in an adaptive, personalized, optimal, guidable, just-in-time fashion, and wholistic fashion—an approach that directly meets the goals of smart learning environments.

Acknowledgements We gratefully acknowledge funding support from NSERC Canada, Athabasca University, and Shastri Indo-Canadian Institute for this research.

References

- Abdous, M., He, W., & Yen, C. J. (2013). Using data mining for predicting relationships between online question theme and final grade. *Educational Technology and Society*, 15(3), 77–88.
- Allen, J. F., Guinn, C. I., & Horvitz, E. (1999). Mixed-initiative interaction. *IEEE Intelligent Systems*, 14(5), 14–23.
- Angelo, T. A., Cross, K. P. (1993). *Classroom assessment techniques: A handbook for college teachers*, (2nd ed.). San-Francisco: Jossey-Bass.
- Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of CSCL*, 2(2–3), 150–190.
- Bienkowski, M., Brecht, J., & Klo, J. (2012). The learning registry: Building a foundation for learning resource analytics. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (pp. 208–211).
- Boekaerts, M., Corno L. (2005). Self-regulation in the classroom: A perspective on assessment and interventions. *Applied Psychology: An International Review*, 54, 199–231.
- Brooks, C., Winter, M., Greer, J., & McCalla, G. (2004). The massive user modelling system. In *The Proceedings of the 7th International Conference on Intelligent Tutoring Systems, Maceio, Brazil* (pp. 635–645).
- Butler, D. L., Schnellert, L., & Higginson, S. (2008). *Fostering agency and co-regulation: Teachers using formative assessment to calibrate practice in an age of accountability*. Paper presented at the American Educational Research Association, April 2008.
- Cross, K. P., Angelo, T. A. (1988). *Classroom assessment techniques: A handbook for faculty*. Ann Arbor: University of Michigan, National Center for Research to Improve Postsecondary Teaching and Learning.
- Dicheva, D., Mizoguchi R., & Greer J. (2009). *Semantic web technologies for e-learning*. IOS Press, ISBN 978-1-60750-062-9.
- Essa, A., & Ayad, H. (2012). Improving student success using predictive models and data visualisations. In *Proceedings of the ALT-C 2012 Conference* (pp. 58–70).
- Ferguson, R. (2012). Learning analytics: Drivers, developments and challenges. *International Journal of Technology Enhanced Learning (IJTEL)*, 4(5/6), 304–317.
- Ghidini, C., Pammer, V., Scheir, P., Serafini, L., & Lindstaedt, S. (2007). APOSDLE: Learn@ work with semantic web technology. In *Proceedings of the International Conference on Knowledge Management (I-Know '07)*, Graz, Austria.
- Hung, J. L., Hsu, Y. C., & Rice, K. (2012). Integrating data mining in program evaluation of K-12 online education. *Educational Technology and Society*, 15(3), 27–41.
- Jovanović, J., Gašević, D., Torniai, C., Bateman, S., & Hatala, M. (2009). The social semantic web in intelligent learning environments-state of the art and future challenges. *Interactive Learning Environments*, 17(4), 273–308.
- Kim, M., & Lee, E. (2013). A multidimensional analysis tool for visualizing online interactions. *Educational Technology and Society*, 15(3), 89–102.
- Kosba, E., Dimitrova, V., Boyle, R. (2005). Using student and group models to support teachers in web-based distance education. In *Proceedings of the 10th International Conference on User Modeling, Edinburgh, UK* (pp. 124–133).
- Manyika, J., Chui M., Brown, B., Bughin, J., Dobbs, R., & Roxburgh, C. (2011). *Big data: the next frontier for innovation, competition, and productivity*. Report by McKinsey Global Institute. Retrieved January 10, 2014, from http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
- Mazza, R., Dimitrova, V. (2003). CourseVis: Externalising student information to facilitate instructors in distance learning. In *Proceedings of the International Conference in Artificial Intelligence in Education (AIED 2003)* (pp. 279–286).
- Mazza, R., Dimitrova, V. (2004). Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th World Wide Web Conference, NY, USA* (pp. 154–161).

- Mazza, R., Milani, C. (2005). Exploring usage analysis in learning systems: gaining insights from visualisations. In *Proceedings of the Workshop on Usage analysis in learning systems at the 12th International Conference on Artificial Intelligence in Education, Amsterdam, The Netherlands* (pp. 65–72).
- Medeiros, F., Gomes, A. S., Amorim, R., & Medeiros, G. (2013). Architecture for social interactions monitoring in collaborative learning environments as a support for the teacher's awareness. In *Proceedings of International Conference on Advanced Learning Technologies (ICALT)* (pp. 123–127).
- Rao, S., Kumar, V., Hatala, M., Gašević, D. (2007). Mixed-initiative interfaces to recognize regulate and reflect programming styles. In *Proceedings. of the Workshop on AI for Human Computing at 20th International Joint Conference on Artificial Intelligence (IJCAI), India* (pp. 71–78).
- Romero, C., Ventura, S., & Garcia, E. (2008). Data mining in course management systems: Moodle cases study and tutorial. *Computers and Education*, 51(1), 368–384.
- Rountree, N., Rountree, J., Robins, A., Hannah, R. (2004). Interacting factors that predict success and failure in a CS1 course. *ACM SIGCSE Bulletin*, 36(4), 101–104.
- Shakya, J. (2005). *Knowledge engineering and knowledge dissemination in a mixed-initiative ontological framework*. MSc thesis, Simon Fraser University, Canada.
- Shum, S. B., & Ferguson, R. (2012). Social learning analytics. *Educational Technology and Society*, 15(3), 3–26.
- Watson, C., Li, F. W. B., & Godwin, J. L. (2013). Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *Proceedings of International Conference on Advanced Learning Technologies (ICALT)* (pp. 319–323).
- Wood, E., Zivcakova, L., Gentile, P., Archer, K., Pasquale, D. D., & Nosko, A. (2013). Examining the impact of off-task multi-tasking with technology on real-time classroom learning. *Computers and Education*, 58(2011), 365–374.
- Zinn, C., and Scheuer, O. (2006). Getting to know your student in distance-learning contexts. In *Proceedings of the 1st European Conference on Technology Enhanced Learning* (pp. 437–451).



<http://www.springer.com/978-3-662-44446-7>

Smart Learning Environments

Chang, M.; Li, Y. (Eds.)

2015, XII, 219 p. 61 illus., Hardcover

ISBN: 978-3-662-44446-7