ASQ: Active Learning with Interactive Web Presentations and Classroom Analytics

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Vasileios Triglianos

under the supervision of
Prof. Cesare Pautasso

August 2018

Dissertation Committee

Prof. Mária Bieliková	Slovak University of Technology, Bratislava, Slovakia
Prof. Pierre Dillenbourg	École polytechnique fédérale de Lausanne, Lausanne, Switzerland
Prof. Matthias Hauswirth	Università della Svizzera italiana, Lugano, Switzerland
Prof. Mehdi Jazayeri	Università della Svizzera italiana, Lugano, Switzerland

Dissertation accepted on 27 August 2018

Research Advisor	PhD Program Director
Prof. Cesare Pautasso	**Prof. Dr. Walter Binder**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Vasileios Triglianos
Lugano, 27 August 2018

To my beloved family, close friends and Dimitris Floros

Knowledge which is acquired under compulsion obtains no hold on the mind.

Plato

Abstract

Today it is commonly accepted that the effectiveness of instructional technology, measured as improvement over the learning outcomes of students, is highly correlated with the learning environment, the applied pedagogies and the goals of the participants of the learning process. The often applied passive lecture model and the undirected use of modern technology such as student laptops and smartphones has been associated with increased levels of inattention and poor student performance. To address this, our work is focused on creating instructional technology for orchestrating traditional post-secondary brick-and-mortar classrooms of computer science education where active learning is the pedagogy of choice. We promote active learning during interactive lectures featuring retrieval practice with open practice question types of all answer depth formats(recognition, cued/free recall) directly integrated with slides. These turn student devices from a source of distraction to a learning affordance. Moreover, we take advantage of modern real-time Web technologies and machine learning techniques to allow timely and effortless gathering, assessment and classification of all student responses and activity during lectures thus tackling issues of scale with extrinsic classroom activities.

This dissertation introduces `ASQ`, a Web application for increasing teacher awareness by (i) turning student's devices from distraction tools to learning affordances; (ii) facilitating the application of active learning with the use of question types of various formats and depth; and (iii) utilizing real-time data analytics to facilitate the collection of students submissions, accelerate feedback cycles and infer student behaviors dynamics. With `ASQ` lecture slides are transformed into an interactive social playground for knowledge construction where students experiment with the presented material (individually or collaboratively), answer questions and continuously give feedback about the lecture quality. We reinforce the role of teachers as the driver of classroom activity by providing them with information to follow the progress of the learning process, spot learning gaps or misconceptions early and provide feedback when needed. We begin by focusing on the engineering aspects of such an application and discusses in depth

how to architect interactive presentations for the Web and design an extensible set of active learning question types for live audiences. Next, we move to the educational technology domain and combine several longitudinal case studies in real-world computer science courses involving hundreds of students, which showcase the potential of a data-driven approach to infer students dynamics and design more engaging lectures, with student and instructor evaluation studies of `ASQ` to confirm its the suitability as lecturing tool in the modern classroom.

Acknowledgements

First and foremost, I am extremely grateful to my advisor Prof. Cesare Pautasso and I feel truly lucky for working with him. Cesare not only believed in me and gave me the opportunity to complete this PhD, but is also a great mentor. Cesare would patiently listen every time there was a problem and provide, but not force, critical advice and insights in his calm demeanor. He handled pressure with grace and helped immensely when deadlines where looming. He also taught me a great deal in various aspects of the researcher's life: from writing in a better way to thinking more abstractly and presenting my work in an efficient way. It was very convenient for the topic of this thesis that Cesare loves to teach and have a real impact on his students' learning. He was a perpetual source of ideas and support which fundamentaly shaped the contribution of this dissertation. Thanks again Cesare!

Special mention goes to Prof. Matthias Hauswirth, an excellent educator, who was a big inspiration of this work that was based on his project: Informa. Moreover Matthias, has been following my PhD closely all these years providing very insightful feedback that has found its way in this work. I am also very happy he has agreed to be a part of the dissertation committe.

I would be remiss not to thank Prof. Mária Bieliková of the Faculty of Informatics and Information Technologies at the University of Bratislava. She helped tremendously with the work in this dissertation in various capacities. She used our software to give her lectures and conduct experiments – in some cases utilizing specialized equipment that was crucial for some of the work in this thesis – and provided us with invaluable feedback. Moreover, as an expert in technology enhanced education, she has mentored me a lot through this dissertation. Last but not least, I am fortunate to have her as part of my dissertation committee.

I am also very glad to have Prof. Mehdi Jazayeri and Prof. Pierre Dillenbourg in my dissertation committee, since they are accomplished educators with great expertise on teaching. Prof. Jazayeri's influential work at teaching and encouragement to pursue research on my project were great sources of influence. Prof. Dillenbourg introduced *classroom orchestration*, a model for the role of in-

structional technology that shaped this work.

A big thanks goes to Prof. Alessandro Bozzon and Prof. Claudia Hauff of the Web Information Systems group at the Delft University of Technology. They were the first people outside of my faculty that expressed interest in the work of this thesis and the were very generous to provide me with an internship and trust me to run our software in their courses. On top of this, they are excellent mentors and great researchers that helped advance my experimental design skills, critical thinking and writing a lot. Thank you guys, it was a pleasure collaborating with you.

A great portion of this dissertation was carried out thanks to my colleagues and friends Vincenzo Ferme, Achille Peternier, Daniele Bonetta and Evangelos Niforatos who helped me in various stages of this work by giving me advice, sharing resouces helping me in experiments, collaborating with me or just hanging out with me and having a coffee.

Another great portion was contributions of students that I had the pleasure to supervise and collaborate with in various student projects and BSc/MSc theses: Margarita Grinvald, Max von Bülow, Jacques Dafflon, Georgios Kokosioulis, Zhenfei Nie, Marco Racazzini, Emrah Signorini, Valerie Burgener, Gustavo Graziani and Daniele Lo Preiato.

Dimos and Popi, my parents, have given me unlimited love, support and guidance all these years. Their contribution is priceless (even more than this dissertations' :-)).

Vassilis Papapanagiotou has not only been one of my closest friends over the years but he was the one who suggested, researched and organized our doctoral studies in Switzerland. Many thanks to you my friend, I feel very lucky to have you in my life.

Finally, there is a very special person who has been a great influence and a mentor in my adult life: Dimitris Floros. Together with Cesare and Vassilis, they form a triplet of people whom without I would have never managed to start, let alone complete, my PhD program.

Contents

Contents **xi**

List of Figures **xvii**

List of Tables **xxi**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Research Questions 3
1.2.1 Web-based Interactive Presentations tool to Utilize Students' Devices as a Tool for Active Learning and Monitoring of Classroom Interactions 4
1.2.2 Mining Browser Events to Infer High Level Student Attention States 4
1.2.3 Cues for (in)attention 5
1.2.4 Comparing Student Visual Attention Data with Activity Indicators obtain from the browser 5
1.2.5 Practice Question Temporal Placement Strategies in Lectures 5
1.3 Contributions 5
1.4 Ethics 6
1.5 Summary and Outline 7
1.6 Publication Overview 9

2 State of The Art **11**
2.1 The Role of Technology in Education and Modern Pedagogies . . . 12
2.1.1 Modern Pedagogies 14
2.1.2 Active Learning 15
2.1.3 Classroom Orchestration 15
2.1.4 Retrieval Practice and Answer Depth Format 17
2.2 Challenges of Introducing Active Learning in Traditional Lectures 17

2.2.1 The Importance of the Lecture Format . . . 17
2.2.2 In-class Questions . . . 19
2.2.3 The Dual-Process Model of Cognition and Cognitive Load 20
2.2.4 The Effects of Cognitive Load . . . 21
2.2.5 Teacher Bias . . . 23
2.3 Digital Technology in the Classroom . . . 23
2.3.1 Student attention . . . 25
2.3.2 Audience Response Systems . . . 28
2.3.3 Plugin Systems for Web Applications . . . 30
2.4 Summary . . . 31

3 ASQ 33
3.1 Teachers's Point of View . . . 35
3.1.1 The Power of the Web . . . 35
3.1.2 Scaling Active Learning and Analytics . . . 36
3.2 Students' Point of View . . . 37
3.3 Question Types . . . 38
3.4 Question Evaluation from Students . . . 40
3.4.1 Setting . . . 40
3.4.2 Results . . . 41
3.5 Summary . . . 43

4 ASQ Plugin System 47
4.1 Requirements . . . 47
4.2 The Asqium Plugin System . . . 50
4.2.1 Design Goals . . . 51
4.2.2 Server-side Plugins . . . 53
4.2.3 Front-end Components . . . 53
4.2.4 Communication and Message passing . . . 54
4.2.5 Persistence . . . 57
4.2.6 Security and Plugin Isolation . . . 58
4.2.7 Deployment and Release Engineering . . . 59
4.2.8 Crafting a plugin for ASQ . . . 59
4.3 Summary . . . 63

5 ASQ as an Educational Study Platform 65
5.1 Case Study: Measuring Student Behaviour Dynamics in a Large Interactive Classroom Setting . . . 65
5.1.1 Threats to Experimental Validity . . . 66

5.1.2 Methodology & Use Case 67
5.1.3 Results 70
5.2 Usability and Perceived Workload 75
5.2.1 Course Overview 75
5.2.2 Usability 76
5.2.3 Perceived Workload 77
5.3 Summary 78

6 Case Study: Inferring Student Attention 81
6.1 Introduction 81
6.2 From Low-level Events to Attention States 82
6.3 ASQ Deployment & Data Collection 85
6.4 Analysis 86
6.5 Summary 93

7 Exploratory Study: Comparing Eye-tracking Gaze Data With the Visibility ASQ Activity Indicator 95
7.1 Experimental Setup and Methodology 96
7.1.1 Course 96
7.1.2 Experimental Setup 97
7.1.3 Methodology 98
7.2 Results 99
7.3 Discussion 102
7.4 Summary 103

8 Case Study: Practice Question Strategy 107
8.1 Methodology & Use Case 107
8.1.1 Question Difficulty 109
8.2 Results 110
8.3 Summary 111

9 Conclusions 113
9.1 Summary of Contributions 114
9.1.1 Answers to Research Questions 115
9.1.2 A Novel Paradigm to Interleave Study Material Slides with Questions 116
9.1.3 A Web Component library of Interactive Question Widgets 117
9.1.4 Using a Custom URL Schema to Reuse Questions into Presentations 117

9.1.5 Design and Implementation of a Plugin System to Facilitate creation of Active Learning Question Types 117
9.1.6 Design and Implementation of 14 Question Types to practice Computer Science 117
9.1.7 Mining Browser Events to infer Higher level Engagement States and Offer Insights on Student Behavior Dynamics . 118
9.1.8 Insights Towards Making Lectures More Engaging 118
9.2 Limitations 119
9.3 Future Work 119
9.3.1 Incremental Improvements 120
9.3.2 Monitoring Instructors 120
9.3.3 Outside The Classroom 121
9.3.4 Submission Clustering 122
9.3.5 Adaptive Teaching and Predicting Analytics 123
9.4 Final Remarks 123

A ASQ User Guide 125
A.1 ASQ Classroom Cycle 125
A.2 ASQ Authoring Workflow 127
A.3 Interactive Presentation Markup 131
A.3.1 <asq-exercise> 132
A.3.2 Welcome element 133
A.3.3 Teacher Utilities 135
A.3.4 Common elements 137
A.4 Question Types 137
A.4.1 Text (<asq-text-input-q>) 138
A.4.2 Multiple Choice (<asq-multi-choice-q>) 140
A.4.3 Code (<asq-code-q>) 143
A.4.4 Highlight (<asq-highlight-q>) 145
A.4.5 CSS Select (<asq-css-select-q>) 150
A.4.6 Live JS(<asq-js-eval-q>) 152
A.4.7 JS Function<asq-js-function-body-q> 154
A.4.8 SQLite (<asq-sqlite-q>) 156
A.4.9 Rate (<asq-rating-q>) 158
A.4.10 Classify (<asq-buckets-q>) 160
A.4.11 Order (<asq-order-q>) 164
A.4.12 HTML Fiddle<asq-fiddle-q> 166
A.4.13 Java (<asq-java-q>) 169
A.4.14 Diagram (<asq-diagram-q>) 172

B ASQ Software Architecture **175**
B.1 Design and Implementation 175
B.1.1 ASQ Architectural Overview 175
B.1.2 Data Model 176
B.1.3 Core Functionality 177

Bibliography **187**

Web References **205**

Student Projects **209**

Figures

2.1 Current (left) and proposed stack (right) 18

3.1 ASQ in the classroom: most students' laptops are connected and focused on a slide that contains the results of an interactive question. 34
3.2 A high level schematic view of the main concepts of ASQ. These basic concepts form the data model of Figure B.1 35
3.3 Boxplot of ASQ question type ratings. 44
3.4 Temporal evolution of rating for the ASQ question types. 45

4.1 ASQ architecture with the plugin system 49
4.2 Plugin structure: Back-end modules and Front-end Web components 52
4.3 Proxy object used as the entry point façade for the plugin contributed business logic to access the core APIs 53
4.4 Hook lifecycle: registration and chained invocation 54
4.5 Viewer (left) and Presenter (right) views of a highlight question . 60

5.1 Sequence of events for three example viewers during the first 15 minutes of Lecture 2, *HTML*. Each vertical line represents a slide change — there are 29 slides shown in total, one of which is a question slide. 68
5.2 Distribution of slide engagement (computed per student) across all lectures. A value of 1 (0) indicates that ASQ was always (never) visible during class time. 72
5.3 Temporal evolution of connected and slide engaged students during the *HTTP* lecture. Evidence of distracted behaviour in the second half of the lecture. 73
5.4 Temporal evolution of connected and slide engaged students during the *CSS* lecture. No clear slide engagement trend can be observed. 73

5.5 Temporal evolution of connected and slide engaged students during the *Javascript* lecture. Evidence of increasing slide engagement trend. Three interactive non-question slides are marked in green. 74
5.6 Boxplot of SUS scores for student responses 76

6.1 Two example progressions of inferred attention states during the course of a single 90-minute lecture (specifically: Web Security). The dark-grey areas represent slides with interactive exercises (6 in total), while the light-grey vertical bars indicate slide transitions. While *Student 1* starts off highly attentive, *Student 2* is inattentive from the start of the lecture. 84
6.2 ASQ in the classroom: most students' laptops are connected and focused on the slide material being explained. 85
6.3 SQLite question from Lecture 1, *Advanced SQL*. It comprises a text editor (left) to write and execute SQL queries on an in-browser database instance, and a results pane (right) to visualize the query results. 86
6.4 Connected and Focused activity indicators for all the sessions . . . 90
6.5 Inferred student attention state for all the sessions 91
6.6 Attention Focus Variation: how many students have changed the focus of attention during lecture (moving sum of attention breaks and recoveries using a window of 30 seconds). 92

7.1 Example of a highlight question posed in the lectures *Prolog 2a* and *Prolog 2b*. 98
7.2 State combination of fixation on/off screen and visible/hidden ASQ window for all students of the lecture *Prolog01b*. **NULL** values are displayed for completeness. 102
7.3 State combination of fixation on/off screen and visible/hidden ASQ window for all students of the lecture *Prolog02a*. **NULL** values are displayed for completeness. 103
7.4 State combination of fixation on/off screen and visible/hidden ASQ window for all students of the lecture *Prolog02b*. **NULL** values are displayed for completeness. 104
7.5 State combination of fixation on/off screen and visible/hidden ASQ window for all students of the lecture *Prolog03a*. **NULL** values are displayed for completeness. 104

7.6 Pie charts with all the combinations of fixation on/off screen and visible/hidden ASQ window. Each pie chart contains the mean for all lectures. 105
7.7 Pie charts with all the combinations of fixation on/off screen and visible/hidden ASQ window for various *conditions*. Each pie chart contains the mean for all lectures. **Legend**. *WQ*: Slides that contained questions. *NQ*: Slides without a question. 105

8.1 A temporal view of question strategies - each dot represents when a question was asked. 109

9.1 Where learning is happening 122

A.1 The ASQ classroom cycle. 126
A.2 Rendered impress.js slide for the code of Listing A.1 128
A.3 Rendered reveal.js slide for the code of Listing A.2 129
A.4 The "upload" page of ASQ. On the top there is the drag-and-drop area that accepts ZIP archives. On the bottom, the command to upload a new presentation via *curl* is displayed. 130
A.5 A pop-up window displaying the reupload command for an existing presentation. 130
A.6 Left: a presentation in the "presentations" page with the *"play"* button that starts the presentation highlighted (green color). Right: a presentation that is "live" in the "presentations" page with the *"stop"* button that stops the presentation highlighted (red color) . 131
A.7 Example of an <asq-exercise>. 134
A.8 Example of an <asq-welcome> element. 134
A.9 Example of an <asq-canvas> element. 136
A.10 Example of an <asq-text-input-q> question type element. . . . 139
A.11 Example of an auxiliary <asq-text-input-q-stats> element that displays statistics for the submissions of a <asq-text-input-q> element. 139
A.12 Example of two <asq-multi-choice-q> question types. 141
A.13 Example of a <asq-multi-choice-q> that contains an <asq-option> with the is-other attribute set. 142
A.14 Example of an <asq-code-q> question type. The presenter view is showing 3 submitted answers. 144
A.15 Example of an <asq-highlight-q> question type. 145
A.16 Heatmaps in <asq-highlight-q> presenter view. 147

A.17 The <asq-highlight-q> editor GUI. Here shown in progress of creating the question of Listing A.10 148
A.18 HTML markup output for an <asq-highlight-q> question that includes the solution 149
A.19 Example of an <asq-css-select-q> question type. 151
A.20 Example of an <asq-js-eval-q> question type. 153
A.21 Example of an <asq-js-function-body-q> question type. 155
A.22 Example of an <asq-sqlite-q> question type. 157
A.23 Example of two <asq-rating-q> question types. 159
A.24 Example of the audience view for the <asq-buckets-q> question type. 161
A.25 More advanced example of an <asq-buckets-q> question type element. 162
A.26 Aggregated statistics displayed in the Presenter View for the <asq-buckets-q> question of Figure A.24 163
A.27 Example of an <asq-order-q> question type element. 165
A.28 Example of an <asq-fiddle-q> question type element. 167
A.29 Audience view example of seeding the editors of an <asq-fiddle-q> question type element. 168
A.30 Example of an <asq-fiddle-q> question type element. 170
A.31 Directory structure two support two Java exercises for the <asq-java-q> question type named ‘exercise1’ and ‘exercise2’. 171
A.32 Example of an <asq-diagram-q> question type element. 173

B.1 Overview of the relationship between the entities of ASQ’s data model. For higher level overview see Figure 3.2. 178

Tables

1.1 Overview of Research Question (RQ) we aim to answer, the sections we briefly discuss them and chapters where they are addressed in this dissertation 3

1.2 Summary of the main contributions with pointers to the chapters and the publications where they are presented. Publication numbers are pointers to Table 1.3 for easier access. 6

1.3 Summary of the publications with work relevant to this dissertation. 10

2.1 Related work overview: column 2 reports the number of lectures and the length of each lecture (in minutes). Column "Simulated" indicates whether the experiment was conducted in a simulated classroom (✓) or a natural classroom (✗) setting. Students' behaviours can be determined as follows: assigned condition (behaviour determined by experimental condition assigned), self-reports (students report on their behaviour/distraction), human observers (observers sit behind students and observe them) or online activity logging (through a proxy server or ASQ in our work). The "Learning performance measurement" reports how students were evaluated on their learning performance. 26

2.2 ASQ Feature comparison of tools that support functionality related to our work. 29

3.1 ASQ question types 39

3.2 Overview of retrospective sessions on ASQ and its question types. **Legend**. **#SQ** number of students that participated in the questionnaire. **#QT** number of question types that were evaluated. **QTN** name of the question types under evaluation. 41

3.3 Overview of retrospective sessions on ASQ and its question types. **Legend**. **MMC** Mean rating for *Multiple Choice*. **MT** Mean rating for *Text*. **MJSFB** Mean rating for *JS function*. **MCSS** Mean rating for *CSS Select*. **MHL** Mean rating for *highlight*. **MCL** Mean rating for *Classify*. **MSQL** Mean rating for *SQLite*. **#SQ** **#SQ** number of students that participated in the questionnaire. 43

5.1 Lecture overview, reported in temporal order of delivery. **Legend**. **#Q**: number of questions in the slides. **#STU**: number of students in the classroom. **%ASQ**: the percentage of students in the classroom connected to ASQ. $\rho(\boldsymbol{S}, \boldsymbol{Q})$: Spearman rank order correlation between *slide* and *question* engagement scores. 69

5.2 Mean SUS score of student responses per lecture session. "–" in the number of SUS responses denotes retrospective sessions where there was no lecture. 77

5.3 Mean SUS score of student responses per lecture session † Cockpit view did not work for next slide preview 78

6.1 Overview of activity indicators based on browser events. 82

6.2 Modeling student attention based on activity indicators. Activity indicators are binary, ✓ represents True, and - represents False. . 83

6.3 Overview of the three ASQ lecture sessions each given by one of two instructors (identified as I1 and I2). For each session, the number of students participating, the number of exercises (per type) and the number of ASQ low-level browser events logged are listed. 87

6.4 Linear correlation coefficient (significant correlations at the $p < 0.05$ level are marked †) between time and number of students exhibiting a particular activity indicator (top part) or one of a set of inferred attention states (bottom part). 88

6.5 Correct vs incorrect answers ordered by time of question for Lecture 2 93

7.1 Overview of the lectures in the case study. Lectures are reported in temporal order of delivery. The lectures were divided into two separate sessions (e.g., *Prolog 1a* and *Prolog 1b*), i.e., the same content was lectured to two different groups of students. **Legend**. **#ASQ**: number of students connected to ASQ. **#Q**: number of questions in the lecture slides. **#A**: number of total answers in the lecture. **%SQ**: the percentage of students connected to ASQ and answering at least one question (≥1 question submitted). **%75Q**: percentage of students connected to ASQ and submitting an answer to at least 75% of the questions in the lecture. **%AQ**: percentage of students connected to ASQ and submitting an answer to all questions. **%SN**: percentage of students' initiated inputs to the questions that resulted in no submit. 96

7.2 Number of complete session records for each session. 97

7.3 Summary – for all lectures and all slides – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen the ASQ window *visible* and ASQ window *hidden* states captured by ASQ. 99

7.4 Summary – for slides without a question in all lectures – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen the ASQ window *visible* and ASQ window *hidden* states captured by ASQ. 100

7.5 Summary – for slides with questions in all lectures – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen during the ASQ window *visible* and ASQ window *hidden* states captured by ASQ. 100

7.6 Summary – for different conditions across all lectures – of the percentage of time that the students' gaze during a lecture is fixated is on or off the screen the ASQ window *visible* and ASQ window *hidden* states captured by ASQ. **Legend**. *WQ*: Slides that contained questions. *NQ*: Slides without a question. We report the following conditions: *input*: learners were giving an input to a question in the current slide. *submitted*: learners had submitted an answer to the questions of the current slide. *idle*: users had more than 10 seconds of inactivity during the current slide. 101

8.1 Lecture overview, reported in temporal order of delivery. These are the same lectures summarized in Table 8.1 with a focus on metrics pertinent to this study. **Legend**. **INS**: instructor ($I1$ or $I2$). **QS**: the question strategy (b=burst, i=increasing and u=uniform). **#EQ** and **#HQ**: number of easy and hard questions in the slides. **%SQ**: the percentage of students connected to ASQ and answering at least one question ($\geq$1 question submitted). **%AQ**: percentage of students connected to ASQ and submitting an answer to all questions. $\rho(\boldsymbol{S},\boldsymbol{Q})$: Spearman rank order correlation between *slide* and *question* engagement scores. **%CA**: percentage of correct answers. Correlations significant at the $p < 0.01$ level are marked with a †. 108
8.2 Overview of correlations between *slide* and *question* engagement, computed separately for each instructor, each question strategy and the four combinations of instructor and strategy. N is the number of items to correlate. %CA is the percentage of correct answers. Correlations significant at the $p < 0.01$ level are marked with a † 110

B.1 Overview of generic Web browser events monitored by ASQ. 183
B.2 Overview of ASQ-specific Web browser events. 184
B.3 Overview of server generated events of ASQ. 185

Chapter 1

Introduction

From the first educational manuscripts to massive open online courses (MOOCs), technology has played an important role in education. Educational content can be shared and be experienced in many formats that range from text forms, to video lectures to augmented reality 3D visualizations. Modern technology has managed to scale course audiences to unprecedented numbers allowing knowledge penetration in groups where time, location, money or language would, in other times, be a barrier. Finally, analytics can help with various tasks involved in the learning process, such as designing better courses, grading assignments, identifying learning obstacles, or adapting learning material to the student skills and abilities.

However, technology has also been associated with neutral or detrimental effects to learning. Digital devices (most often laptops and smartphones), though desired tools by students in a higher education classroom, have in the past been shown to serve more as distractors than supporters of learning. One of the reasons is the often undirected nature of the devices' usage. In our work, where we focus in technology aimed toward traditional brick-and-mortar computer science lectures, technology has frequently served as distraction: student's devices have invaded classrooms and compete with the instructor for attention forcing students to multitask which degrades the learning experience: it is often the case where we find our students browsing social networking websites or playing games while trying to absorb lecture material and answer the instructor's questions. Moreover, the complexity added by introducing multiple educational tools often lead to increased context switching that can also act as a distraction and leads to wasting valuable teaching time. Finally, technology has not been able so far to scale more effective forms of learning, such as mastery learning, active learning or practice testing, to lecture audiences of real-world classrooms

of hundreds of students despite the generally acceptable positive impact of these teaching styles. It is obvious to us through our experience with teaching computer science, that in the “practice versus theory” [100] challenge that we face in the curriculum of a modern software engineer, traditional lectures lean towards the “theory” part; leaving “practice” (active learning) for post-class activities which can significantly delay deep learning of the material and make it harder for teachers catching misconceptions before they take root.

The current technology-enhanced education landscape can be segmented based on three criteria: whether the educational content is delivered online or offline; whether the interactions between the instructors and the students are performed in a synchronous or asynchronous way and finally, whether the students are co-located with the teacher. In this dissertation, we follow a "best of breed" approach and argue that traditional lectures in a co-located setting should be augmented and enhanced by using tools and techniques that have been originally proposed and developed for online education. Our work is motivated by the “bring your own device" trend where more and more students attend classes with their laptops, tablets and smart phones. In particular, we aim to investigate how we can repurpose the student’s devices to encourage students to engage in active learning and use analytics mined from their devices to gather cues on their behavior to enhance teacher awareness by fine-grained monitoring of engagement levels, providing timely feedback and improving future editions of the courses.

1.1 Problem Statement

In the context of computer science lectures in post-secondary traditional brick-and-mortar classrooms, students have limited opportunities to give the teachers feedback for their level of understanding of the study material. This can be an even bigger obstacle for shy and unsure students. Classroom time, complexity and teacher bias of the answers render the cost of posing active learning in-class assignments that promote effortfull retrieval to all students and assessing them prohibitive, in spite of the fact that they are very effective in promoting learning retention [114; 76; 83]. Moreover, the use of modern technology in the classroom – such as laptops and smartphones – distracts students with off-task behavior that has been associated with negative student performance [77; 188; 155; 93].

Our thesis is that the problem of scaling active learning to the aforementioned settings can be tackled by utilizing modern Web technology in ways that help engage students with practice question types and support teacher awareness by

real-time insights on the students level of comprehension and behavior.

1.2 Research Questions

We posit that modern technology can help drive active learning engagement in modern brick and mortar classrooms and provide insights in student behavior through the use of analytics that can help measure and improve engagement. To prove our claim, we seek to answer the following Research Question (RQ):

RQ1 *To what extent can a Web-based, privacy-preserving teaching tool be reliably used to enable active learning, infer student behavior dynamics, assess the attention and learning outcomes of students and provide insights to improve teaching?*

To tackle the broad nature of this RQ with tasks that are easier to quantify, we focus on work on the following refined RQs listed in Table 1.1.

Research Question (RQ#)	**Chapter(s)**	
RQ1.1 *To what extent can a Web-based, privacy-preserving teaching tool be reliably used to assess student behavior dynamics and learning outcomes of students from their interactions with the tool?*	subsection 1.2.1	5,6
RQ1.2 *To what extent can students' attention be inferred from their interactions with a Web-based interactive presentations application that facilitates in-lecture active learning questions practice?*	subsection 1.2.2	6
RQ1.3 *Which type of interactions are most correlated with (in)attention?*	subsection 1.2.3	6
RQ1.4 *How does student gaze behavior as recorded through the use of eye-tracking relates to activity indicators generated from browser events?*	subsection 1.2.4	–
RQ1.5 *Does the practice question strategy — questions deployed uniformly across the entire lecture or with bursts of several questions at the same time – have any impact on student engagement and student learning?*	subsection 1.2.5	8

Table 1.1. Overview of Research Question (RQ) we aim to answer, the sections we briefly discuss them and chapters where they are addressed in this dissertation

1.2.1 Web-based Interactive Presentations tool to Utilize Students' Devices as a Tool for Active Learning and Monitoring of Classroom Interactions

It is generally accepted that active learning leads consistently to increased learning outcomes compared to frontal lectures especially in Science, Technology, Engineering and Mathematics (STEM) disciplines [142; 76]. However, scaling active learning to large cohorts of students remains particularly difficult task due to constrains associated with time and workload that are imposed on the instructor. In this thesis, we explore whether the use of technology can alleviate some of this burden. One reason that learning technology has failed to (consistently) demonstrate increased learning outcomes is because it did not introduce a leap in pedagogy [179]. Our thesis, in agreement with the community, is that using technology in manners that facilitate the application of desired pedagogies can have a positive impact. In the context of our work, computer science education in large post-secondary classrooms, it is a common phenomenon for students to bring their smart devices in the classroom and at the same time the introduce a new source of distraction. The maturity of Web Technology, its decoupled open access model of resources and the ubiquitous presence of browsers in smart devices make it an ideal candidate to offer its services in traditional post-secondary classrooms in an effort to scale active learning, increase directed use of technology towards the learning goals and help instructors orchestrate activities around it.

In this dissertation, we design and build a Web tool for use in real-world classrooms with the aforementioned goals. The tool supports streaming lecture slides that may contain active learning practice question types to the students devices. We deployed the tool in real-world courses, sometimes with hundreds of students and investigate to what extent students engage with the tool to follow slides or practice active learning by utilizing its analytics capabilities.

1.2.2 Mining Browser Events to Infer High Level Student Attention States

Building on the previous subsection, we investigate the potential of browser events to extract semantic meaning of the students actions in the platform and create a model for student attention that can help instructors adapt their teaching and design more effective lectures. Once again, we trial our approach in real-world classrooms.

1.2.3 Cues for (in)attention

After evaluating our approach in creating high level attention states from browser events, we quickly realized that we could combine them with findings from the state of the art in student attention to create a profile for student interactions that correlate with increased attention. We analyze recorded lectures with our tool in similar settings as the previous subsections to spot patterns in student engagement.

1.2.4 Comparing Student Visual Attention Data with Activity Indicators obtain from the browser

Students being engaged with their devices for on-task activities such as answer practice questions or following the lecture slides accounts for just a fraction of their total activities during class time. It would be interesting to explore how much of their visual attention, as reported by eye-tracking apparatus, is devoted to interacting with the tool we described in subsection 1.2.1 and to what extend to they visually engage with their screen based on the current activity performed in the tool.

1.2.5 Practice Question Temporal Placement Strategies in Lectures

One of the advantages of having a tool that interleaves questions with answers is the ability to collect engagement metrics regarding the practice questions across a lecture session. We utilize this metrics from real-world courses, to gather insights in a mostly overlooked topic of practice question sessions: their temporal placement strategy.

1.3 Contributions

The overarching goal of this dissertation is to contribute the design, development and evaluation of ASQ, a tool for delivering lectures in conventional, small-to-large sized (between 10-300 students) brick-and-mortar classrooms and university lecture halls that aims to drive active-learning and serve as a source for data-driven insights in the areas of teacher awareness, student engagement, student attention and lecture design.

Our main contributions include a novel paradigm to interleave study material slides with questions, a library of interactive question types, a plugin system for creating active learning question types and techniques to infer student engagement and attention from Web browser events. To demonstrate the effectiveness of our design, we present qualitative studies in various computer science in real-world settings. In a similar fashion, the ability of the tool to gather insights in student behavior dynamics is reflected in the results of a series of quantitative studies in similar settings. A summary of the contributions in this dissertation is listed in Table 1.2.

#	**Contributions**	**Chapters**	**Publications**
1	A Novel Paradigm to Interleave Study Material Slides with Questions	3	[1, 2]
2	A Web Component library of Interactive Question Widgets	3	[3]
3	Design and Implementation of 14 Question Types to practice Computer Science	3	[4, 5, 6]
4	Design and Implementation of a Plugin System to Facilitate creation of Active Learning Question Types	4	[3]
5	Using a Custom URL Schema to Reuse Questions into Presentations	4	–
6	Mining Browser Events to infer Higher level Engagement States and Offer Insights on Student Behavior Dynamics	5,6	[4, 5, 6]
7	Insights Towards Making Lectures More Engaging	5,6,7	[4, 5]

Table 1.2. Summary of the main contributions with pointers to the chapters and the publications where they are presented. Publication numbers are pointers to Table 1.3 for easier access.

1.4 Ethics

The cornerstone of our research case studies in real-world classrooms with human subjects (instructors and learners) is to respect the privacy of- and provide equal opportunities to learning to- those involved. Our experimental protocols and study designs have been designed with these principles in mind. The data

gathered by the software tool that we present in this dissertation were stored in authentication protected databases using state of the art encryption. In all but one setting student participants were anonymously logged into the system. Despite that, their answers could still identify them for which reason we restrict access to them to only the author and the instructors of the respective courses. Under no circumstances were the grades given during the lecture affected by the data collected during the experiments.

In the setting where we used eye tracking technology to track the attention of the students (see subsection 7.1.1)we collected *personal and identifiable data* for which we informed the subjects. These data have been stored in an authentication protected server accessible only from the researchers conducting the experiment.

We also conducted usability studies using encryption protected and fully anonymized Google Forms [goo] questionnaires.

Finally, we conducted workload testing with instructor subjects, that they also contained personal and identifiable data. We used forms in paper format which are privately stored by the author.

1.5 Summary and Outline

This dissertations comprises 8 chapters and 2 appendices. Below we provide a brief description of each chapter and appendix:

- **Chapter 1 - Introduction**. In the first chapter we introduce our vision on how the modern brick-and-mortar classroom can benefit from using technology to promote active learning and data-driven insights on student behavior dynamics. We define the motivation behind this work, the research questions that are answered through our thesis, and we present an overview of the challenges, the contributions and the related publications of this work. We also describe the ethics that govern our research case studies.

- **Chapter 2 - State of the Art.** In the second chapter we do a synopsis of the research areas this work draws on. We examine topics like digital technology in the classroom, student attention, practice questions during lectures, the role of technology, teacher bias and class response systems.

- **Chapter 3 - `ASQ`.** Here we present the primary contribution of this work: `ASQ`, a Web-based application for interactive presentations. We discuss the

impact it has in the classroom and provide results of quantitative and qualitative usability evaluations collected from students in real-world courses.

- **Chapter 4 - ASQ Software Architecture.** In this chapter we introduce the *asqium* the plugin system of ASQ which is the basis of all ASQ question types. We list the desired properties of the system and shed light on how we tackled the design challenges we faced to achieve the required extensibility. We present implementation details and examples for interested parties that wish to implement a plugin to create additional interactive question types.

- **Chapter 5 - ASQ as an Educational Study Platform.** This chapter shifts the focus of this dissertation from engineering aspects to instruction technology research. We discuss the requirements of real-world classes in terms of privacy and equal conditions of participation for students. We report on a case study where we define metrics for engagement based on Web telemetry data captured by ASQ. We also report on a qualitative evaluation survey for the usability of the system gathered from students and for the imposed workload gathered by interviewing instructors. Our findings in this chapter make a strong case for using ASQ as a research platform.

- **Chapter 6 - Case study: Inferring Student Attention.** Our second case study is presented in this chapter. It builds on the event-driven approach of chapter 5 to mine high level attention states. Our results show that ASQ's insights are aligned with the literature and can be used to non-intrusively infer student attention in real-time.

- **Chapter 7 - Case study: Practice Question Strategy.** Here, we present our last case study that utilizes the question and analytics capabilities of ASQ to compare different strategies for introducing practice questions during a lecture.

- **Chapter 8 - Conclusions.** In the final chapter, we summarize the answers to the research questions posed in the current chapter and the contribution of this dissertation. We also discuss any limitations concerning our approach. We reflect on the implications that derive from using active learning coupled with Web-backed analytics. We conclude by looking into the future and sharing our vision on the modern classroom and some of the next steps on how to achieve it.

- **Appendix A - ASQ User Guide.** In the first appendix of this work we take a pragmatic look into using ASQ. We discuss the main features of the tool

and present how to use ASQ in the classroom to support active learning and how to author interactive content. We finish this chapter by presenting the interactive elements of ASQ, placing a strong focus on the question types. We present the rationale behind their creation and usage examples.

- **Appendix B - Software Architecture.** In the second and final appendix we focus on the design and implementation challenges of ASQ beyond its plugin model. We list our requirements and design decisions, its data model and how critical aspects of the application function. We also present its analytics capabilities upon which a lot of the work presented in this dissertation was based on.

1.6 Publication Overview

Part of the work in this dissertation has been published in international peer-review conferences. Table 1.3 displays a classification by whether the publications had an engineering or educational context.

	#	Publications
Web Engineering	1	**Vasileios Triglianos** and Cesare Pautasso. ASQ: Interactive web presentations for hybrid MOOCs. *In Proc. of the 22nd International Conference on World Wide Web*, WWW '13 Companion, pages 209–210, New York, NY, USA, 2013. ACM.
	2	**Vasileios Triglianos** and Cesare Pautasso. Interactive scalable lectures with ASQ. *In Proc. of the 14th International Conference on Web Engineering (ICWE 2014)*, pages 515–518, Toulouse, France, July 2014. Springer.
	3	**Vasileios Triglianos** and Cesare Pautasso. Asqium: A javascript plugin framework for extensible client and server-side components. *In Proc. of Engineering the Web in the Big Data Era: 15th International Conference*, pages 81–98, Rotterdam, The Netherlands, June 23-26 2015, Springer.
Education Technology	4	**Vasileios Triglianos**, Cesare Pautasso, Alessandro Bozzon, and Claudia Hauff. Inferring student attention with ASQ. *In Proc. of the 11th European Conference on Technology Enhanced Learning (EC-TEL)*, Lyon, France, September 2016. Springer.
	5	**Vasileios Triglianos**, Sambit Praharaj, Cesare Pautasso, Alessandro Bozzon, and Claudia Hauff. Measuring student behaviour dynamics in a large interactive classroom setting. *In proc. of the 25th International Conference on User Modelling, Adaption and Personalisation (UMAP 2017)*, Bratislava, Slovakia, July 2017. Springer.
	6	**Vasileios Triglianos**, Martin Labaj, Robert Moro, Jakub Simko, Michal Hucko, Jozef Tvarozek, Cesare Pautasso, and Maria Bielikova. Experiences using an interactive presentation platform in a functional and logic programming course. *In Proc. of the 7th International Workshop on Personalization Approaches in Learning Environments (PALE)*, Bratislava, Slovakia, July 2017. ACM.
	7	Michal Hucko, Peter Gaspar, Matus Pikuliak, **Vasileios Triglianos**, Cesare Pautasso, and Maria Bielikova. Short Texts Analysis for Teacher Assistance during Live Interactive Classroom Presentations. *In Proc. of the World Symposium on Digital Intelligence for Systems and Machines (DISA)*, Košice, Slovakia, August 2018. IEEE. (to appear)

Table 1.3. Summary of the publications with work relevant to this dissertation.

Chapter 2

State of The Art

Lectures in todays' form have been around since the medieval times [26; 78] and remain the predominant mode of instruction since universities were founded in Western Europe [53]. Despite the advantages of having an expert disperse information on a subject, the passive model of presenting large amounts of information to audiences has several shortcomings: there is not a practical way for instructors to ensure that students are intellectually engaged with the material due to their passive attitude; students' attention wanes quickly; it is hard to assess the level of comprehension of the audience; learning is inefficient due to the lack of active retrieval of the taught material; and finally, lectures presume that students absorb material at the same pace. These shortcomings are more pronounced for topics that require higher orders of thinking such as application, analysis, synthesis, or evaluation, for teaching motor skills, or for influencing attitudes or values. One solution to improve the situation is to limit lecture segments [82] or complete eliminate them (flipped classroom [21]) and engage in more efficient forms of learning such as discussion [154], Problem-based learning (PBL) [97][1] or active learning [76]. However, we think that learning in a classroom has social and psychological benefits for students. In the words of Genie Black, "The traditional classroom has the major advantage of face-to-face interaction between the student and educator as well as between the students themselves. Students derive motivation from the teacher as well as from the other students" [23]. In this dissertation we aim to to promote the shift from the frontal lecture paradigm (monologue) to interactive bi-directional presentations and discussions (dialogue) by introducing active learning in modern brick-and-mortar classrooms.

[1] Prince argues that some aspects of the Problem-based learning process may be considered active learning [142].

According to the classroom orchestration process, a term coined by Fischer and Dillenbourg [73], in our work we focus on providing solutions to a number of issues associated with the *core*, *envelope* and *infra* categories of extrinsic activities that take place in our target classroom. We face three main challenges that inhibit the classroom orchestration process: 1. the intrusion of *student devices* in the classroom which further exacerbate the engagement and attention problems; 2. teachers catering to the needs of only a small portion of the classroom – and thus, gathering and incomplete and often misleading picture of the level of comprehension of the classroom – due to *teacher bias* and *cognitive load* constraints; and – related to the previous challenge – 3. *scaling* the delivery, assessment and feedback for active learning questions to larger audiences.

In the following sections we give an overview of the effectiveness of technology in education and argue that the role of technology is to support suitable education pedagogies. We present the pedagogy that our work focuses in – active learning – and an overview of classroom orchestration. We make the case for in-class practice questions to promote active learning and discuss the various formats of retrieval. Next, we move to review four challenging issues for scaling active learning in the modern classroom: 1) the lecture format, 2)in-class questions, 3) cognitive load and 3)teacher bias. Moreover, we discuss the presence of digital technology within modern classrooms. We discuss how smart devices have invaded lectures and ways to utilize them for on-task activities: streaming lecture slides, answering questions and gauging student attention. To paint a full picture of how technology has been used in a directed way to engage students, we take a look into Audience Response Systems (ARS), which we consider to be predecessors of our work and other pertinent technological solutions. Finally, we turn to a different engineering topic and look into plugin systems for Web applications as they will be a vital part in implementing the extensible set of active learning question types during our work.

2.1 The Role of Technology in Education and Modern Pedagogies

It has been argued that educational technology has not fulfilled the great expectations for better learning outcomes. One of the reasons is that, until recently, new technology did not introduce a leap in pedagogy. As Valdez et al. [179] observe for the first waves of technology in education: “educational software was mostly textbooks presented in electronic print formats”. The focus, still, was not on

the learner. The control was just moved from the teacher to computer programs [179]. Critics suggested that the learning process was dehumanized [174] and ignored the "learner satisfaction, self-worth, creativity, and social values" [178, p. 37]. The situation seemed likely to improve with the shift of learning technologies towards the principles of constructivism in the 1980s. Nevertheless, this step was deemed unsuccessful to significantly restructure the established learning environments, perhaps due to the extended time computer-based activities needed to complete [189] and the additional material they imposed to the central curriculum [3]. In the late 1980s and early 1990s the disappointment kept growing with Clark maintaining the position that media will never influence learning [46].

The culmination of the information age with the emergence of the Internet, enabled students to access a plethora of information resources in a cost effective manner. According to Earle [66], the new ways in which the Internet allowed people to communicate, share and access information and collaborate triggered calls for transforming the curriculum, fitting certain technological applications into the existing curriculum, and even fitting the curriculum to the computer. By some observers the new Web-based experiences were perceived as a quantitative increase in information accessibility and availability, rather than a qualitative change in the curriculum [79; 66]). Hart in [89] holds a more extreme view: "What we need from educational technology is forms of knowledge which may lead to understanding, rather than information overload and confusion" (retrieved from Ramsdens book "Learning to teach in higher education" [133, p. 152]).

Albirini [7], building on the work of Illich [98] that views the current educational system as product of the industrial age, suggests that the main reasons educational technology has failed to meet its alleged potential are: *a*) lack of a theoretical frame that would justify the incorporation of education technology in the curriculum; This leads to *b*) attempts to fit educational technology, a product of the information age, in the incompatible, prevailing industrial system of education that is governed by different assumptions; and finally *c*) unwillingness and even resistance for change on behalf of those involved in the educational process (teachers, educational institutions, policy makers etc.). The proposed solutions are to either leave the current educational system as is, or to "avail the potential of the new technology after thorougly restructuring education and schools, as remnants of the industrial age, into a new paradigm and institution". Taking the above into consideration, our work is focused on using technology as the enabler of a pedagogy, active learning [29], though addressing issues in the orchestration process of post-secondary computer science classrooms. In the next subsection

we discuss modern pedagogies, we introduce active learning and present a brief overview of classroom orchestration concepts and how they pertain to our work.

2.1.1 Modern Pedagogies

A current trend is to view instructional technology as an enabler of pedagogies [68]. Constructivist and modern pedagogies such as Problem based learning, Active learning, Spaced Learning, Enquiry Based Learning, Project Based Learning and Peer Tutoring have the potential to be greatly facilitated by technology [96; 151]. Students become the center of the learning process and teachers the facilitators. The characteristics of these pedagogies include tracking individual progress, inquire, experimenting, discussing, working in groups, producing artifacts and engaging with the real world. Teachers are required to monitor and assess the learning process of students, which includes reflection on taught concepts, experimentation with phenomena and creation of artifacts. This has a great impact on the cognitive load (see subsection 2.2.3 for an in-depth discussion) of teachers and it becomes practically impossible to accomplish these tasks effectively in populous classrooms, especially when taking into consideration the tendency of teachers to provide more favorable conditions of learning for subgroup of the students (see subsection 2.2.5 where we discuss teacher bias).

In a similar fashion, Mayer argues that to avoid more disappointments from instructional technology we should used in ways that are grounded in research-based theory ([118]) and [80] states that the key to better learning outcomes is thinking system change knowledge, pedagogy and technology in an integrated way. In the 2000's there is a shift of focus in instructional technology that places the learner at the center of the learning process. The result of these change was a new body of research in how individuals construct knowledge. New instructional technology motivating deep learning, based on cognitive psychology and neuroscience. Application of cognitive load theory in education technology ([117])

All in all, learning technology in isolation of the other entities that comprise the learning process, namely teachers; learners; pedagogies and educational environment, seems to have little to no effect in the learning outcome. In the next subsections we discuss a pedagogy associated with positive outcomes [124; 142; 76] in STEM fields and how we envision using it to teach computer science in the classroom: active learning.

2.1.2 Active Learning

Active Learning is an umbrella term for learning (in the classroom) where students engage with the learning process more directly (actively) and generally entails some sort of active retrieval of the taught concepts. In the words of Bonwell and Eison, active learning is “a method of learning in which students are actively or experientially involved in the learning process and where there are different levels of active learning, depending on student involvement" [29]. While some faculty argues that learning is inherently active, in active learning, activities like collaborative work, discussion of materials while role-playing, debate, engagement in case study, or producing short written exercises are placed at the center of the learning process [29]. Active learning is a natural fit for Science, Technology, Engineering and Mathematics (STEM) disciplines [124; 142; 76] where deep learning of the topics is desired. It is a well-established fact that active learning, in all its various forms, is effective in increasing students' learning performance compared to the traditional lecture setup where students are passive recipients of information [142]. Freeman et al. conducted a meta-analysis of 225 studies across the STEM fields [76] found not only an increased effect size by .47 SDs in performance on examinations and concept inventories but also that the probability ratio for failing a course deacreased by 1.95 compared to traditional lectures. Given its importance, we have set as our mission to enable every student to engage in active learning during lecture sessions in computer science topics through answering practice questions. A great part of our contribution focuses on overcoming limitations in time and workload that are imposed by delivering, assessing and monitoring this active learning process.

2.1.3 Classroom Orchestration

Classroom orchestration refers to how a teacher manages in real-time multi-layered activities in a highly constrained ecosystem. These levels include individual activities, teamwork and class-wide sessions [6]. Technology is viewed as a means that support teachers in coordinating these activities. The teacher is viewed as the ‘conductor‘ who makes the ultimate decisions based on feedback (often provided by technology) on how the intrinsic and extrinsic activities of a learning scenario progress in the classroom.

In a publication [56] that greatly influenced our work, Dillenbourg elaborates on the continuum between intrinsic and extrinsic activities and describes five categories of activities that require orchestration:

1. *Core activities* for example the topics to recite in a lecture;

2. *Emergent activities* are activities that pre-designed but whose outcome is unpredictable and this require real-time elaboration on behalf of the teacher. An example is "debriefing" session where students reflect on what they learn;

3. *Envelope activities* refer to all the activities that are part of established school practices, such as note taking or reporting student progress to parents.

4. *Extraneous events* refer to unscheduled events whose occurrence is outside the control of the teacher, like a loud noise or a student missing for part of a lecture. While teachers cannot predict such events, they still have to handle them in ways that benefit the learning of students; and

5. *Infra activities*. These are activities that are not meaningful for the educational scenario but necessary to run it. Examples include logging to computers and finding the right documents.

Next, he outlines five types of constraints to take into consideration when designing systems to support classroom orchestration: *time*, *curriculum relevance*, *discipline*, *assessment*, *energy* and *space* constraints. In the context of our work – supporting active learning in classrooms with large number of students – we wish to help teacher to better deal with the time, discipline, assessment and energy constraints. We need to be able to deliver question types, aggregate submissions as fast as possible to save precious lecture time and energy. We wish to provide an easy and summarized ways for teachers to access submissions from all students so that they can assess their level of understanding as effortlessly as possible. Finally, we wish to monitor student activity and how they engage with the material to minimize distractions and adapt teaching.

Dillenbourg extracts five design principles, four of which that are relevant to our work: *control*, *visibility*, *flexibility* and *minimalism* (the fifth being *physicality* which we did not address in this work). Control refers to teacher being the driver of all activities in the classroom. Their "decision should supersede any system decision, because the former has contextual information while the latter does not". Visibility refers to the exchange of information on the activities that happen in the classrooms by all actors to all actors. flexibility refers to the ability to adapt on-the-fly a learning scenario based on feedback cues from monitoring students or reacting to extrinsic events and constraints. Finally, minimalism caters to the cognitive load of the teachers by suggesting to use only the information necessary for them to make informed decisions and avoid introducing unnecessary extra complexity. In chapter 3 we will discuss how our technological solution tries to adhere to these principles.

2.1.4 Retrieval Practice and Answer Depth Format

The *testing effect*, taking tests on studied material to promote active learning as opposed to using tests only as an evaluation tool, is beneficial to retention and leads to better scores in final tests [17; 50; 88; 94; 121; 122; 184]. As Carrier and Pashler demonstrated, the influence of successful retrieval, that is, recalling items from memory, "has beneficial effects for later retention above and beyond the effects due to merely studying the item" [41]. This result has been confirmed in various other replicating studies(for example [150; 4; 102]).

There are three types answer depth format for questions that test memory recall that are in line with the general model of memory recall: 1. *recognition*. Subjects are given a list of potential answer with the task to recognize the correct one; 2. *cued recall*. Subjects are given a number of cues to help them retrieve and construct the right answer from memory; and 3. *free recall*. Subjects need to remember from memory the right answer without any help. The rough equivalent of these formats in educational terms are multiple choice, short answer, and essay questions respectively [114].

Of big interest for our work is the fact that there is strong evidence which suggests that successful retrieval under the more effortfull *free recall format* for quizzes (such as short text quizzes or programming tasks) perform better in terms of learning outcomes compared to *cued recall*, which performs second best, or *recognition formats* (such as "fill the gaps" or multiple choice quiz types) [114; 83; 127; 120; 106]. Mixing theoretical explanations with interactive exercise activities benefits long-term retention as it helps students to reflect on the taught material (which is still in their working memory) and prevent misconceptions from taking root (as teachers can give early feedback). This is one of the pillars of our work, in which we aim to scale in-lecture practice active learning questions with a strong focus in the free and cued recall formats to large audiences.

2.2 Challenges of Introducing Active Learning in Traditional Lectures

2.2.1 The Importance of the Lecture Format

A typical lecture involves an instructor presenting the reviewed subject(s) while navigating through slides, drawing in a whiteboard or interacting with artifacts pertinent to the subject. The slides can include a number of multimedia or interactive features such as images, videos and hyperlinks. Periodically, they may dis-

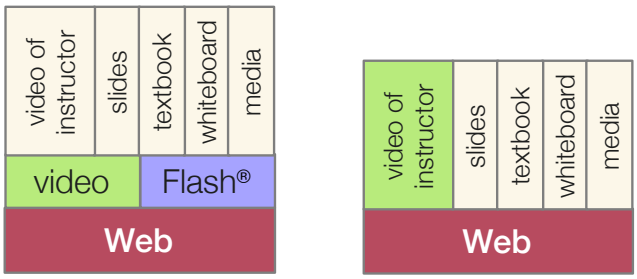


Figure 2.1. Current (left) and proposed stack (right)

cuss with audience members often to clarify misconceptions and/or query their comprehension on the taught concepts. A major limitation of this format is that not all students have the equal amounts of concentration on the presentation at any given time or comprehend educational content at the same pace.

An interesting lecture format that solves this problems can be found in lectures that are intented to be consumed asynchronously such as, traditional MOOC lectures, which are delivered in form of videos. These videos typically include instructors presenting the reviewed subject(s) while navigating through slides, scrolling through a textbook, drawing on a digital whiteboard or commenting on displayed media [186]. Often, during a presentation, after the instructor elaborates on a specific concept, a pop quiz follows that aims to evaluate the level of comprehension from the audience. One benefit of asynchronous lectures is the ability for students to revisit sections of the lecture at their discretion. This solves the problem deriving from the assumptions that all students process content at the same pace. We believe that allowing students of colocated synchronous lectures to browse slides can also have the benefit of decreasing off-task behavior like browsing social media websites or playing games (see section 2.3).

However content authoring for asynchronous lectures requires multiple software stacks: slides creation software, video capture and editing software; the hosting platform that serves the video lectures, the quizzes and the assignments. Slides are not browsable anymore; they are part of a monolithic format that makes it very hard to make corrections. Even for simple adjustments on a slide, video recapture is necessary. Figure 2.1 visualizes the current situation and our proposed solution for both synchronous and asynchronous lectures: fully use the modern Web platform for both authoring and delivering lectures and gathering feedback.

Another limitation, as Cooper and Mehran [49] point out, is the need to augment video presentations with interactive activities like algorithm visualizations, programming practice environments and editable coding visualizers. This ap-

plies not only to video lectures, but to lectures in general: Bonwell and Eison [29] suggest as way to further increase engagement the integration of brief demonstrations or short, ungraded writing exercises. They also suggest to modify the lecture format by injecting introspective discussion sessions so that students can reflect on the material or allocating time slots for students to write what they remember. Our approach is to follow a variation of the latter where we pose practice questions to students. We discuss this in more detail in subsection 2.1.4 and subsection 2.2.2. By simply featuring interactive sections, lectures are not more effective in terms of learning outcomes. A study conducted in six public university campuses [32] finds "no statistically significant differences in learning outcomes between students in the traditional and hybrid-format sections" and that "there is no compelling evidence that online learning systems available today—not even highly interactive systems, which are very few in number—can in fact deliver improved educational outcomes across the board, at scale, on campuses other than the one where the system was born, and on a sustainable basis. This is not to deny, however, that these systems have great potential". We postulate that two reasons play an important role: first, a lot of the quizzes belong to the recognition answer depth format of answer depth (for example multiple choice questions) that has limited retention benefits compared to the cued or free recall formats (also discussed in subsection 2.1.4); and second, there is the lack of two psychological/social components: 1) a physically-present teacher who is motivating students and monitors their behavior and learning progress to provide timely feedback and catch misconceptions early. This allows teachers to intervene in subtle ways that benefit the learning scenario of the course; and 2) other students that help treat learning as a team effort and interact in ways (discussion/group work) that "improves thinking and deepens understanding" [45]. In our work, in which we target the co-located classroom domain, we lift these restrictions through incorporating active learning question types such as live-programming questions types or drawing diagrams and by utilizing real-time aggregation of answers and monitoring of student engagement levels to increase teacher awareness.

2.2.2 In-class Questions

In classes with a large number of participants, active learning is most often associated with questions that allow immediate automated feedback, such as multiple-choice questions or (relevant in our use case) programming questions evaluated through unit testing. This is due to time constraints and the cognitive load involved in assessing open-ended questions. In our work we attempt to introduce

more cued/free recall question types due to the benefits involved. One of our strategies is to introduce question types that enable *live programming*, i.e., the ability to modify running programs (in our case inside a browser) [75]. A major benefit of live programming is the instantaneous application of changes in the static code a program to the dynamic behavior of its executable, thus "providing continuous feedback of program content to programmers.". This can offload a great portion of the task of evaluating correctness to students and encourage them to experiment.

Recently, Weinstein et al. [182] explored the benefit of quiz spacing in the classroom: in a within-subject design (45 students) two spacing strategies — interspersed and at-the-end — were compared with each other. Similarly to our own case study presented in chapter 8, the questions (five per lecture) in the interspersed condition followed directly the slides containing the necessary information, while in the at-the-end condition all questions were placed at the end of the lecture. The achieved quiz scores in the interspersed condition were found to be significantly higher than in the at-the-end condition. This difference in learning performance though vanished when the students were tested one last time nearly three weeks after the last lecture.

2.2.3 The Dual-Process Model of Cognition and Cognitive Load

Studies in teacher cognition and teaching performance in K-12 classrooms have identified consistent challenges and patterns that are congruent with the predictions of the dual-process model of cognition (e.g., [72]). Cognitive load, a phenomenon associated with the controlled conscious component of the model, is the cause for many of these challenges.

According to the dual-process theory of cognition, human performance is produced by controlled and automatic processes. Controlled processes occur in the working memory, function more slowly and require more effort than other processes. Automatic processes occur without intention; are not subject to conscious monitoring; utilize few, if any, attentional resources; and happen rapidly [125; 183]. These processes operate independently but intersect at certain points [15; 16; 54; 158; 165]. Information processing occurs simultaneously in both pathways. When two pathways generate conflicting outcomes, the conflict is mediated in the working memory [31] which results in a performance drop.

Cognitive load is an index that refers to the total amount of mental effort being used in the working memory [166]. Sweller [168] identified three types

of cognitive load: instrinsic, extraneous, and germane. Extraneous cognitive load (allocation of cognitive resources for less relevant goals) occupies working memory at the expense of intrinsic and germane load, depriving the individual of cognitive resources that could benefit target performance [42; 169].

While in this dissertation we focus on creating and evaluating a system that imposes an acceptable (self-reported) workload on instructors, the next section delves into the effects of cognitive load to provide a context for discussing some future steps on how address them in subsection 9.3.2.

2.2.4 The Effects of Cognitive Load

Automaticity

Routines for specific tasks within a domain of expertise become highly reliable and require less concentration to perform as individuals practice their skills thus imposing lower levels of cognitive load [9]. In the education domain, novice teachers report feeling overwhelmed [40; 105; 185] since they have not yet developed automaticity for routines that need to be performed in the classroom. Experienced teachers that perform the same routines, require significant less concentration and experience lower levels of cognitive load which "provide[d] them with the intellectual and temporal room necessary to handle the dynamic portions of the lesson" [110]. In contrast, the increased cognitive load imposed on novices led to a slower and more effortful perfromance that limited the dynamic allocation of mental resources.

High levels of cognitive load deplete working memory which cannot be used by dual-process cognitions, like situation assessment and attribution. As a result, these mental processes may rely almost entirely on their nonconscious components and operate without conscious monitoring [15; 71; 175]. Described by Feldon as "the double-edged sword of automaticity" [72], in such situations there may be an increase in performance but also errors. Performance gains are observed when the automated schemas of these processes have yield optimal performance without conscious monitoring [18; 19; 86]. Errors occur when the representations are stereotypic schemas that incorporate irrelevant or insufficient information. The conscious monitoring would either prevent the inappropriate application of these schemas or adapt them to align with individual circumstances. Increased bias in responses under high cognitive constraints have been documented in [84] and [27]. Additional studies found that individuals formed and stored evaluations of events that were more stereotypic when processed under high cognitive load than they would be with additional conscious

monitoring [43; 54].

Regarding teaching, there are studies that demonstrate biased inferences due to insufficient allocation of cognitive resources (e.g., [87; 115]) and increased racial bias associated with elevated levels of extraneous cognitive load and speeded time regardless of the articulated beliefs or intentions of their participants [27; 55; 70; 135; 181]. Experienced teachers demonstrate lower expectancy bias [13] than novices, using evaluation procedures that are are more resistant to interference presumably due to higher levels of automaticity (which limit the number of consciously mediated decision points during which expectancy could play a role) and lower cognitive load (which allows monitoring processes to function unimpeded).

Cognitive defaults

The substitution of intended actions with less effortful, automatic and potentially destructive alternatives under increased levels of cognitive load is known as a cognitive default [47; 131; 167]. According to [72]: "The cognitive default hypothesis similarly suggests that teachers' older, more reinforced knowledge is more likely to be used under the cognitive load imposed by an actual classroom, because it is less effortful than newer knowledge. Therefore, the actions of new teachers are likely to default to the manner in which they were taught rather than the way they were trained".

Although there is a lack of studies that examine changes in teacher's behaviour under the lens of cognitive load theory [72], there are studies the describe novice teachers who revert to their more familiar, pretraining models of teaching [128; 145]. This suggests that the old, less effortful automatic schemas acquired during their own schooling experiences [112] are more likely to be used under cognitive load instead of the teaching methods that the teachers were trained.

Ironic processes

Wenger [181] described another interesting effect of increased extraneous cognitive load during attempts for self-control: ironic mental processes. In such instances, trying to achieve the intentional outcome results, ironically, to a counter-intentional outcome. The premise is that controlling one's mind involves an operating process which is a conscious component and a monitoring process which is an automatic component. The monitoring process scans for undesirable mental contents that are inconsistent with the intended state. Upon detection, it initi-

ates the operating process which will try to attend to it and effortfully (since its a controlled, working memory dependent process) modify it. When the individual experiences exhaustion of working memory resources the operating process fails to function effectively. Consequently, the monitoring process tries to engage it again consuming more working memory and resulting in a feedback loop that not only fails to prevent undesired thoughts or actions from occurring, but may also lead to cognitive defaults.

2.2.5 Teacher Bias

Research shows that teachers in traditional classrooms give students in the top third the greatest attention and students receive the least attention and support (e.g., [28]). Teachers are frequently unaware of the fact that they are providing more favorable conditions of learning for some students than they are for other students. Generally, they are under the impression that all students in their classes are given equality of opportunity for learning. This probably gets worse in bigger classrooms. A number of studies on the biases of teachers in the classroom have indicated that stereotyping occurs with some frequency [148]. For example, studies of the Pygmalion effect or self-confirming bias indicate that teachers' preconceptions of student traits and abilities are better predictors of subsequent evaluations than actual classroom performance [85; 152]. Other biases, including those related to physical attractiveness, have the same effects [20]. In a meta-analysis of studies examining the biases of teachers regarding attractive students, Ritts et al. concluded that, overall, attractiveness had a moderate effect ($d = .41$) on teachers' assessments of students. Measures specific to academic outcomes (e.g., intelligence, future academic performance, etc.) generated an effect size of .36 [148].

2.3 Digital Technology in the Classroom

Wood et al. [188] examined the impact of multi-tasking in lectures in particular through digital means: 145 undergraduate students took three classes of 20 minutes, each followed by a 15-item multiple choice quiz. Students were randomly assigned to one of seven experimental conditions. The learning performance (the percentage of correctly answered quiz questions) was found to be significantly and negatively impacted by multi-tasking (also confirmed in [147]), in particular when it involves highly-engaging social networks such as Facebook.

Closer to our own experimental setups, in one of the seven conditions the

participants were free to use or not use their laptops (i.e. no multi-tasking was forced on them); it was found that those students opting not to use digital technologies achieved a higher learning performance than those that did. A similarly designed study and result was reported in [155], where not only the participants' multi-tasking was investigated but also the impact it had on participants in direct view of the multi-tasker — the distraction to the non-multi-tasking peers was significant and they in turn reached lower test scores than non-distracted peers. A much larger study by Aguilar-Roca et al. [5] across 800 students and 15 lectures did not find detrimental effects to students in the proximity of laptop users as measured in exam grades. As [188; 155], Fried [77] relied on an undergraduate psychology student class to reach the same conclusion through weekly self-reports and the students' test performance. In [93] a binary setup was employed: students were either allowed to use their laptop in class for any activity of their choice or disallowed to use their laptop at all. Students in the "open laptop" condition were found to remember less of the lecture content than students in the "closed laptop" condition. Finally, Ravizza et al. [144] performed the most natural experiment by routing students' Internet traffic in class through a proxy that logged all online activities during class time which was subsequently classified as either class-related or class-unrelated. They found (not surprisingly given past research) that class-unrelated Internet usage (e.g. the use of social networks or emailing) was common among students that chose to use their laptop in class and was negatively related to class performance. More surprisingly, class-related Internet usage did not benefit students, their class performance (usually measured through an after-lecture quiz or final course grades) did not increase over students that did not use their laptops in class.

Despite the varying experimental setups (cf. Table 2.1), the multitude of studies converge on the same conclusion: the use of digital devices in the classroom is not advantageous for students' learning performance due to students' multi-tasking behaviour and the available distractions. Nevertheless, students themselves perceive technology in the classroom as mostly useful instead of distracting [109; 14]. Since the complete ban of technology in the classroom is often not feasible (though less radical ideas such as laptop-free zones within a large classroom have shown promise [5]), we aim in our work to take advantage of technology to establish an additional communication channel between students and lecturers.

Previously, the use of personal response systems[2] ("clickers") has been ex-

[2]Such personal response systems can either be dedicated pieces of hardware or software installed on mobile phones and laptops.

plored as one potential positive use case for interactive technologies (besides laptops) in large classrooms. Mayer et. al [119] found students engaged in in-class multiple-choice question answering through clickers to have higher learning gains than students engaged through the same questions without clickers. Notably, this latter group did also not fare any better than the control group of students who did not receive those in-class questions. Gauci et. al [81] made clicker usage in their classroom voluntary and found students who participated in answering in-class questions this way (in contrast to [119] it was not possible to answer questions without a clicker) to achieve higher exam results than those who did not. Importantly, low-performing students (those with low marks in a prerequisite course) were found to benefit more than mid- and high-achieving students. These results show that a guided and restricted usage of technology can benefit students' learning.

2.3.1 Student attention

Measuring and influencing peoples' state of attention in their workplaces, daily lives and educational settings has been investigated for a number of decades in psychology and pedagogy; in more recent years technological advances have also led to contributions by the human computer interaction and the learning analytics communities [138][141].

Our research focus is in the measuring of students' attention in the post-secondary classroom where forty-five or ninety minute units of teaching are the norm, and thus in this section we narrow our overview to works that have investigated attention in the educational context only. Students' attention during such teaching sessions varies significantly, as shown in a wide range of empirical studies that have either probed students directly for self-reports of attention (or daydreaming and mind wandering) levels [35; 146; 162; 111] or aimed to *infer* [in]attention based on (i) students' behaviour (e.g. their patterns of note-taking [157] or physical signs of inattention such as gazing [103]), (ii) physiological measures such as skin temperature [22], or, (iii) students' levels of knowledge retention [146; 172].

Two important meta-studies [187; 173], published in 2007 and 2013 respectively, not only summarize the current state of knowledge about student attentiveness, but also critically highlight the often contradictory findings — in [187] specifically, the assertion of the 10-15 minute attention span of students is tackled in great detail. The contradictions are generally attributed to the nature of the individual experiments, which are typically conducted on a small number of students taking a class of less than one hour, which may have been specifically

Table 2.1. Related work overview: column 2 reports the number of lectures and the length of each lecture (in minutes). Column "Simulated" indicates whether the experiment was conducted in a simulated classroom (✓) or a natural classroom (✗) setting. Students' behaviours can be determined as follows: assigned condition (behaviour determined by experimental condition assigned), self-reports (students report on their behaviour/distraction), human observers (observers sit behind students and observe them) or online activity logging (through a proxy server or `ASQ` in our work). The "Learning performance measurement" reports how students were evaluated on their learning performance.

	#Lectures (time)	#Students	Simulated	#Exp. conditions	Logging type(s)	Learning performance measurement	Class	Incentive
[188]	3 (20 minutes)	145	✓	7: Facebook — Texting — Natural Technology Use — Word Processing — pen and paper — MSN — email	Assigned condition	15 multiple-choice (MC) questions immediately after the lecture	Research methods, Statistics	$15 or course credit
[155]	1 (45 minutes)	44	✓	2: Multitasking — Non-multitasking	Assigned condition	20 MC questions immediately after the lecture	Introductory Psychology	Course credit
[77]	20 (75 minutes)	137	✗	2: Open laptop — Closed laptop	Weekly self-reports	4 exams and 10 homework assignments	General Psychology	None
[93]	1 (N/A)	44	✗	2: Open laptop — Closed laptop	Assigned condition and voluntary online activity logging	20 MC and open questions immediately after the lecture	Communications	None
[147]	1 (60 minutes)	64	✓	2: Open laptop — Closed laptop	Assigned condition	10 MC questions immediately after the lecture	N/A	$15 or course credit
[144]	15 (100 minutes)	84	✗	3: Class-related Internet usage — Nonacademic Internet usage — No Internet usage	Online activity logging	Final exam	Introductory Psychology	Course credit
[81]	36 (50 minutes)	175	✗	2: Personal response system (PRS) usage — No PRS	PRS logging	Midterm, final exam	Psychology: Control of Body Function	None
[5]	13 (50 minutes)	800	✗	2: Zoned laptop use — Uncontrolled laptop use	Assigned condition	Final exam	Bio 93: DNA to Organisms	None
Our work	14 (90 minutes)	89-319	✗	2: High engagement with `ASQ` — Low engagement with `ASQ`	`ASQ` activity logging	123 questions (MC, highlight, fill-in-the-blank) interspersed in lectures	Web & Database Technology	None

designed for the experiment. Factors which can explain the observed differences include the inherent variability of students' academic interests, instructor styles and means of measuring attention, which are usually not controlled for across experiments [173]. Of the many findings, we list here those which have been observed in several experiments[3]. *F1*: Students' attention drops over the class period [111]; as a consequence, in retention tests students tend to perform better on material presented early on in the class [146]. *F2*: attention breaks occur regularly and increase in frequency as the class progresses [103]. *F3*: As the class progresses, students tend to take less notes [157]. *F4*: the percentage of students attentive to the class varies significantly (depending on class topic, the instructor and the pedagogical tool employed). Between 40% and 70% of students are attentive at any moment during frontal teaching. Attention rises when interactive elements are introduced (discussions and problem solving) [35]. *F5*: immediately after interactive teaching elements, the level of distraction is lower than before the start of the interaction [35; 36].

One common denominator of the aforementioned studies is their lack of technologies to determine students' attention directly or indirectly. Existing technology-based solutions, while enabling real-time insights, are also limited, due to the invasive technologies employed. In [170; 171] EEG signals are recorded to infer students' attention — while accurate, those studies are restricted to either small classroom or lab settings. Sun et al. [164] rely, among others, on facial expressions to detect attention, which, while technologically feasible raises privacy concerns. Bixler et al. [22] find eye gaze and skin conductance and temperature recordings to be indicative of attention. Many of these techniques can only be employed at reasonable cost for a small subset of classes and/or a small subset of students due to their obtrusive nature (examples include physiological markers or minute-by-minute self-reports), issues of scale (e.g., the presence of external observers and the analyses of taken notes), and, the additional cognitive & timely burden placed on students (e.g., through retention tests). Moreover, with few exceptions, e.g. [170], these techniques do not enable lecturers to adapt their teaching on-the-fly, as they are not able to *continuously* determine students' attention *in real-time*; instead students are probed at specific intervals during the lecture or post-lecture data collection and data analyses steps are required.

In contrast, in our work we explore the use of a *non-invasive* and *scalable* technological solution capable of delivering feedback in *real-time* as presented in the case study of chapter 6.

[3] Also for these findings some contradictory evidence exists as well.

2.3.2 Audience Response Systems

Audience Response Systems (ARS), also referred to as Student Response Systems, Personal Response Systems, Classroom Networks [137], Electronic Response Systems, Audience Paced Feedback [139], Classroom Communication Systems and clickers, have been used sparsely for educational and training purposes since 1950s but due to the expense of installing these systems and the lack of evidence for pedagogical benefits they only started to become popular in the 1990's [104]. The most critical feature of ARS is a shared display of student responses, around which students and teachers can focus attention and activity [137]. ARSs today are, or have been, used for many educational disciplines form the physical sciences through mathematics, accountancy, biology, medicine and literature [62].

The most widely reported benefit of ARSs is increased engagement and participation in class [107; 64], sometimes as an indicator of increased student interest and enjoyment of the class [65]. Other advantages include improved classroom discussion about ideas [64], better awareness of student difficulties, both by teachers as well as by the students themselves [62]. Anonymity is important factor that frees students to focus on ideas rather than on who contributed them [52]. Researchers report that the display of a ARS helps students know where they are in relationships to others, and if they notice others are experiencing difficulties students may have the confidence to ask for help or for clarification [64]. The fact that all students pick an answer that then becomes part of an aggregated display puts pressure on students to think about teachers' questions [64; 62; 130]. Students also often report a sense of pride and ownership over their contributions to the display [48].

In our mission to direct usage of digital technology in the classroom to reduce distractions and increased active engagement we aim integrates and improve two complementary product categories: presentation tools (e.g., Keynote [key], Powerpoint [pow], Prezi [pre] or Slides [slia]) and Classroom Response Systems. Table 2.2 features a feature comparison between software ARS. To the best of our knowledge there is no Web-based interactive presentation tool offering a fully integrated experience with slides embedding complex questions. Existing non-integrated tools require a time-consuming switch between the presentation (which may or not be broadcast to all devices) to the dedicated clicker software (which is usually limited to polls using multiple choice or simple questions like the ETH EduApp [eth] or Google Forms). Socrative [soc] is a ARS built around quizzes and games. It exhibits a small set of types of questions and cannot deliver presentations. Informa [91], which is closer to our work than the other systems, is a software clicker implemented in Java that supports an extensible number of

Table 2.2. ASQ Feature comparison of tools that support functionality related to our work.

	Features	**Nearpod** [nea]	**Socrative** [soc]	**Blackboard Collaborate** [bla]	**PropProfs** [pro]	**Informa** [92]	**Glisser** [gli]	**Slido** [slib]
1.	Platform	Web/ iOS/ Android	Web	Web/ Java	Web	Java	Web	Web
2.	Sync Presentations/ Questions	✓/✓	✗/✓	✓/✓	✗/✗	✗/✓	✓/✓	✓/✓
3.	Async Presentations/ Questions	✓/✓	✗/✓	✓/✓	✗/✓	✗/✗	✓/✓	✓/✓
4.	Slide Format	image/ video/ audio	–	Powerpoint	–	–	HTML5	Powerpoint/ Keynote/ Prezi/ PDF
5.	Question Format	Proprietary	Proprietary	Proprietary	Proprietary	Java	Proprietary	Proprietary
6.	Cued/Free Recall Question Types	Text, Draw	Text	Essay, Fill in the Blanks, Opinion, Short Answer	Fill in the Blanks, Essay	Highlight, Infer Type, Text Edit, Graph, Stack and Heap Diagram, Regular Expression	Free Text	World Cloud (open text)
7.	Questions, Presentations as URIs	✗/✗	✓/✓	✓/✓	✓/✓	✓/✓	✗/✓	✗/✓
8.	Send feedback back to students	Teacher can focus and broadcast a specific answer	Share stat results/ correctness (2015)	Share stat results	✗	Share stats, Share results	Share stat results/ correctness	Share stat results
9.	Peer- / self-assessment	✗/✗	✗/✗	✓/✓	✗/✗	✓/✓	✗/✗	✗/✗
10.	Platform host	Company-hosted	Company-hosted	Educational Institution/ Company-hosted	Company-hosted	Educational Institution	Company-hosted	Company-hosted
11.	Share websites within presentation	✓	✗	✓	✗	✓	✓	✓

problem types like text-highlighting problems, coding problems, graph problems.

Very few systems merge the two product categories not only to share slides with the audience in real-time but also to embed polls, questions and interactive content that can be used to gather feedback from the audience. For example, sli.do [slib], Nearpod [nea] – aimed at K-12 education – or Glisser [gli] support only simple question types (polls with a fixed set of alternatives) or free text questions. Blackboard collaborate [24] is a proprietary online collaboration platform that, among other features, offers a dynamic whiteboard that can display and share among participants PowerPoint slides, text, drawings and multiple choice polls. While these are useful to engage the audience, they do not fully fit the requirements for scaling active learning by combining domain-specific, open-ended question types with a fast and accurate way to aggregate and visualize results. All but Informa are offered as a service and are closed source.

2.3.3 Plugin Systems for Web Applications

In contrast to the systems presented in the previous section, in our work we wish enable educators to develop the active learning questions that fit their curriculum and students' needs and be able to exchange material between them to accelerate the adoption of active learning. Thus, our system needs to be extensible and open. We set high goals in terms of requirements for such as a system (for a detailed list see section 4.2) in order to allow powerful, versatile, modular and encapsulated yet easy to develop, maintain and integrate question types. Our design choice is to develop a plugin system for our system that support plugins we can run in both the server and the client (powerful and versatile) and exchange data in real-time. In this section we will look at how others have tackled similar problems to adopt good practices and avoid pitfalls.

Architect [c915], Intravenous [Jac15], Seneca [Rod15] and Wire [cuj15] are all Node.js architectural frameworks that handle well dependency injection and offer ways to declare modules and their dependencies. They are positioned in the application composition layer on top of the npm modules layer. Front-end components are not in the scope of these frameworks.

Our work is heavily influenced by the plugin systems of popular Web-based content management systems. More specifically the hooks construct can be found in similar contexts in Wordpress[MBJ+11], Ghost [WODW15] and Moodle[61] which are two blogging platforms and an open-source learning platform respectively. Moodle has also the notion of event-driven communication between back-end plugins and the core [Mdd15]. In these systems client-server communication between plugin components is performed mainly through HTTP/Asynchronous

JavaScript And XML (AJAX). By default, the back-end components cannot push any data to the front-end without the latter having issued an AJAX request first to pull the data.

Hoodie [Hoo15] is a Node.js framework with CouchDB store technology, whose main goal is to abstract away the back-end to facilitate the job of front-end developers. To accomplish this, the front-end application communicates with the back-end only through the Hoodie Javascript API. Using CouchDB's changes feed Hoodie is always aware of things that happen to the user's data and makes them available via events which allows keeping multiple devices synchronized. It support plugins which have a: *a*) frontend component; *b*) backend component; and *c*) an admin view. Frontend components communicate with back-end components through tasks similar to the client-server event mechanism we describe in subsection 4.2.4. A task is a special object that can be saved into the database from the Hoodie front-end. Front-end plugin components deal only with the Hoodie API and do not have visual entities. Any related markup or CSS styles live in the static assets of the main application outside of the plugin directory. Hoodie thus lacks a way to encapsulate markup and styles for front-end plugins. Hoodie also does not have the concept of hooks. Plugin filenames and directory structure follow strong naming conventions. Hoodie plugins are published and deployed using npm.

2.4 Summary

To summarize, in our quest to scale active learning in the classroom and help teachers with the associated orchestrations activities we discussed the role of technology and highlighted the importance of using it as an enabler of pedagogies and orchestration tasks. We presented some of the deficiencies of the prominent lecture format and we looked in the literature for inspiration on how to improve it. We then presented how previous works (often conducted in simulated classrooms with assigned conditions) have shown that undirected use of digital devices in the classroom leads to distractions and ultimately degrades the students' learning performance. At the same time, the directed use of technology has shown promise. We argue that by using Web technology to enable students' devices to receive rich format slides and be a playground for practice questions we tackle three issues at once: scaling active learning, increasing engagement in learning activities and gaining data-driven insights on student attention and overall engagement. With respect to in-class quizzes, there is little doubt in the literature that interactive classes improve students' learning performance, how-

ever, it is challenging to scale the efficient format of free recall retrieval practice in large audiences. Moreover there is very little work discussing and exploring the benefits of question spacing. Our work adds additional knowledge to this issue and explores to what extent can technology can scale active learning and enable students laptop to function as directed devices in the classroom. In our quest to answer our questions we reviewed relevant audience response systems and highlighted their shortcomings in integrating lecture slides; supporting the more effortful formats of answer depth (cued/free recall); and lack of extensibility. To overcome this limitations we aim to implement a plugin system with focus on creating question types for which we reviewed plugin systems for extensible Web applications.

In the next chapter we will present `ASQ`: a tool whose design is influenced by the topics discussed in this chapter. It aspires to engage students, scale active learning in large classrooms and provide instructors with useful insights on student comprehension and engagement.

Chapter 3

ASQ: A Platform for Interactive Web Presentations

ASQ is a Web-based research application for creating and delivering interactive presentations. ASQ has been designed and developed to increase the lecturer's awareness of the level of understanding in the classroom, and to turn student devices from potential distractors to a novel communication channel. The is designed ASQ to support computer science professors who want to gather real-time feedback from the students while delivering their lectures in a traditional brick-and-mortar classroom but it can be used in any context where presenters require real-time fine grained audience feedback. It is greatly influenced by Informa [91] a Java-based response system that supports multiple question types originally target at teaching Java. The application offers the opportunity to present complex question types in class to hundreds of students and to receive real-time feedback about the students focus, understanding and their performance on the tasks. It has been successfully deployed for more than eight courses in real-world settings to teach computer science courses, which often require learning practical skills and acquiring high level abstraction abilities [101].

Presenters can stream slides to the students smart devices (laptops/smartphones) [243] through a Web client/server architecture. The Web-based nature of the application lowers to a great extent the barriers to set up and use such a system. A link is enough for the audience to join a live presentation. Presenters control the progression of slides from their own device, and changes to the current slide are automatically propagated to all connected student browsers. The slides may contain *quizzes* of various *questions types* that range from simple multiple choice to live programming in JavaScript. With these question types, students are asked to demonstrate their understanding and apply their knowledge in con-

crete scenarios. This way they realize very quickly whether they are capable of solving the given challenges. At the same time, aggregating all students answers allows presenters to assess the level of comprehension of the audience, provide timely feedback and detect misconceptions before they take root. Depending on the question type, student submissions can be clustered to provide a fast and easy to grasp "big picture" of the students' levels of comprehension on a given topic. Figure 3.1 shows ASQ being utilized in a lecture of the "Software Architecture and Design" course of the Spring Semester of 2016 at Università della Svizzera italiana.

Figure 3.1. ASQ in the classroom: most students' laptops are connected and focused on a slide that contains the results of an interactive question.

In our design we took the real-world requirement of students *privacy* into consideration: students submissions are anonymous and the students can participate without providing any login information. This helps an encouraging environments where students can take risks and are not afraid to participate [113]. Figure 3.2 shows the main concepts of ASQ: teacher/students, sessions of presentations, slides, question, answers and answer assessments. In the next two sections we present ASQ from the point of view of the teachers and the students. We then discuss how the is system is typically used within a lecture and the au-

thoring workflow to create content for it. Our final point of focus is the question types of the application. We look at how the interactive presentation markup we developed enables to use full-fledged question types by using simple HTML element. Finally, we list all the question types we developed that have been deployed in real-world courses and discuss evaluation results.

Figure 3.2. A high level schematic view of the main concepts of ASQ. These basic concepts form the data model of Figure B.1

3.1 Teachers's Point of View

3.1.1 The Power of the Web

Installing and learning new software imposes time costs which can disrupt the classroom flow and discourage audience members from using it. A major benefit of ASQ is that it is built using Web technologies and is deployed in the Cloud. ASQ presentations are easily accessible through a modern Web browser, which is a commodity in modern devices.

ASQ manages two different kinds of educational content: lecture *slides* and *questions*. The mature and powerful programming model of the Web empowers

content creators to create interactive, engaging and expressive slides that are suitable for presenting complex topics like algorithms, software architecture and programming. Some algorithmic or architectural concepts can be easier to grasp using parametric visualizations that the viewer can customize, control and advance at their own pace. There are a lot of mature JavaScript-based libraries that enable authors to implement such visualizations that range from simple parametric 2D plots (e.g., *d3.js* by [30], *Chartist.js* by [Kun16] or *Chart.js* [Dow15]) to advanced photo-realistic 3D visualizations (for example *three.js* by [C$^+$10] or *Babylon.js* by [126]).

Another area of Computer Science courses that Web-based slides are a good fit for are topics that involve programming: students can learn and program in code editors embedded directly in the slides. Live-programming is supported for languages that can be evaluated in the browser, such as *JavaScript*, *HTML* and *CSS* or for languages that can be compiled to JavaScript as for example *sql.js [Zak16]*. For the rest of the languages, programs can be executed in serverside sandboxes and results can be reported back to the browser.

Usage of question types that promote active learning in classrooms impose strong scaling demands in terms of distributing the content to a large number of students, evaluating and clustering their submissions and computing real-time analytics. Moreover, question types that enable students to write and execute programs, demand sandboxed environments to run programs separately in order to provide the same robust and secure execution environment to each student. The Web has had similar scalability and robustness requirements in recent years which led to the development of technology for delivering elastic, reliable and cost-effective Web applications [2]. ASQ can scale horizontally by adding more Web servers to serve static content to more users. When needed, analytics computations and sandboxed execution of programs can get offloaded to remote machines whose number can scale according to the number of users and the resource requirements of the workload.

3.1.2 Scaling Active Learning and Analytics

During lectures it's important for students to engage in active retrieval of the taught knowledge to catch misconceptions start creating automatic schemas of knowledge as early as possible [114]. However, as discussed in subsection 2.1.3 and section 2.2, teachers have to perform many different orchestration tasks such as attending to the course curriculum, monitoring the engagement levels of students, or dealing with sources of distraction. This leaves little time for asking questions, aggregating responses, assessing them and giving feedback back to

students. Thus, the time and cognitive demands they impose on the teacher prompts a lot of professors to use exercises mostly for homework material. Without proper technology support, teachers resort to raise of hands or simple clickers based on multiple choice questions. The problem is that the former do not paint an accurate picture of learning retention, while the latter limit experimentation with the taught material to a very superficial level.

ASQ offers the opportunity to present complex question types in class to hundreds of students and to receive continuous real-time feedback about the students' focus, understanding and their performance on the tasks. Questions are not limited to closed types of questions, such as multiple choice, but include many general and informatics related question types that promote active learning and experimentation with the taught material. Example question types include live JavaScript programming; programming Java code and testing it against unit tests; creating Web pages with HTML, CSS and JavaScript; formulating database queries; creating and editing visual diagrams; using CSS selectors [mdn18a]; highlighting text and classifying and rating items. With these question types, students are asked to demonstrate their understanding and apply their knowledge in concrete scenarios. The answers submitted by students are available to the lecturer for review and discussion in real-time. Aggregating all students answers allows presenters to assess the level of comprehension of the audience, provide timely feedback and detect misconceptions before they take root. The combination of multiple question types embedded into slides and clustering of quiz submissions helps teachers avoid time-consuming context-switching, that can also foster distractions among the students. Finally, ASQ features some functionality intended to help instructors orchestrate the answering process. For example, to help with managing the time it takes to complete answering an exercise, each exercise has a progress bar that displays information regarding the number of students that have submitted an answer; the number of students working on the exercise; the number of students have focus on the ASQ window; and the number of students that are idle (see Figure A.7). This progress bar is discussed in more detail in subsection A.3.1.

3.2 Students' Point of View

The passive frontal model of lectures in universities has been the predominant teaching form since 1050 [26]. There are numerous studies that show its ineffectiveness [74; 97; 97] compared to other forms of learning. Students learn better if there's some sort of active learning involved [142; 74]. Another complicating

factor in the modern classroom is the the undirected use of technology. Students frequently exhibit off-task behavior using smartphones, tables and laptops, which are extremely common in modern Universities. An emerging body of research indicates that the multitasking imposed by smart devices has a detrimental effect on various measures of student learning performance, such as attention, recall and exam grades [77; 93; 155; 188; 5]. This phenomenon adds to general trend for inattention exhibited in lectures as discussed in subsection 2.3.1. A potential solution to deal with the distraction that smart devices afford to students is to use technology in a directed manner. For example, Audience Response Systems have been associated with positive learning outcomes [81; 119].

ASQ repurposes student's devices to an active learning tool that promotes experimentation with the taught material. Students can apply their knowledge in concrete scenarios, right after they have been taught a new concept. In computer science lectures, active learning exercises are seamlessly interleaved with lecture material. Students can start live JavaScript programming, drawing diagrams, answering polls, querying databases inside the lecture slides without switching to an external tool. To encourage participation, submissions can be anonymous. ASQ provides various kinds of feedback, that depend on the question type, so that students can continuously track their progress. For example, code in live programming question types is evaluated instantly or multi-choice and text-input question types are marked as correct or wrong. Moreover, the bidirectional channel of communication with the teacher can be used by students to provide unobtrusive and anonymous feedback about the lecture that the teacher can tackle at their discretion.

3.3 Question Types

Active learning in ASQ is enabled through practice question sessions. The questions are embedded into the slides. ASQ currently supports 14 question types of various answer depth format the majority of which are specialized for teaching Web Technologies like HTML, JavaScript and CSS. Table 3.1 lists all the question types of ASQ which will be presented with greater detail in section A.4. They range from recognition depth format – like multiple choice and rating – to free recall depth format questions like JavaScript and SQLite queries. The question types are implemented on top of a plugin system that is explained in detail in section 4.2. This design choice can afford easy extensibility with new question types for different disciplines and teaching contexts.

Table 3.1. ASQ question types

	Name	Description	Type	Assessment	Live Programming	HTML Element	Analytics
1.	Multiple Choice	Multi choice question	recognition	auto	-	`<asq-multi-choice-q>`	horizontal/vertical barchart
2.	Text	Single-line text input (auto assessed)	cued/ free recall	auto	-	`<asq-text-input-q>`	grouped text entries
3.	Code	Code editor with syntax highlighting supporting many programming languages	cued/ free recall	manual	No	`<asq-code-q>`	-
4.	Highlight	Highlight portions of text using the appropriate color	recognition	manual	-	`<asq-highlight-q>`	Heatmap
5.	CSS Select	Find the correct CSS selector	cued/ free recall	manual	Yes	`<asq-css-select-q>`	-
6.	Live JS	Code and evaluate JavaScript code	cued/ free recall	manual	Yes	`<asq-js-eval-q>`	-
7.	JS function	Fill in the body of the JavaScript function	cued/ free recall	manual	Yes	`<asq-js-function-body-q>`	-
8.	SQLite	Run SQLite queries in the browser	cued/ free recall	manual	Yes	`<asq-sqlite-q>`	-
9.	Rate	5-star rating	recognition	auto	-	`<asq-rating-q>`	Average rating
10.	Classify	Classify the label dropping it into the correct bucket (auto assessed)	recognition	auto	-	`<asq-buckets-q>`	Frequency of labels in each bucket
11.	Order	Place items in the correct order	recognition	auto	-	`<asq-order-q>`	-
12.	HTML Fiddle	Code a Webpage using HTML, CSS and JavaScript	cued/ free recall	manual	Yes	`<asq-fiddle-q>`	-
13.	Java	Code and execute Java with JUnit tests	cued/ free recall	auto	No	`<asq-java-q>`	Unit tests
14.	Diagram	Draw ER, UML Class and BPMN diagrams	cued/ free recall	manual	-	`<asq-diagram-q>`	-

3.4 Question Evaluation from Students

To evaluate the experience of students with ASQ questions we conducted small retrospective questions after the lectures of five courses were finished. The questionnaires contained two questions for each question type *QT* that was used in a lecture of a given course:

1. "How would you rate your overall experience with the QT question type". This was a 1-5 points Likert scale where 1 was labeled as "Poor" and 5 as "Excellent".

2. "Let us know any comments/suggestions you have for the QT question type". This was an open text question.

3.4.1 Setting

We posed our questionnaire shortly after the end of five courses where ASQ was deployed and used to deliver the lectures:

1. The 2015/16 version of the course *Software Atelier III*, a compulsory course for 2^{nd} year BSc Informatics students of Università della Svizzera italiana. This was an introductory course in Web Technologies teaching concepts in JavaScript, HTML, CSS3, node.js and MongoDB. The course was followed by 51 students in total. Across the eight course weeks, ten 90-minute lectures were given. 25 students

2. the 2015/16 course *Web and Database Technology* which is presented in detail in subsection 5.2.1;

3. The 2016/17 version of the course *Web Atelier*. This was the successor of the *Software Atelier III* course that catered to the same group of students. The course was followed by 51 students in total. Across the eight course weeks, eleven 90-minute lectures were given together with seven exercise solution feedback sessions.

4. The 2017/18 version of the course *Web Atelier* mentioned above. This time there were 43 students and twelve 90-minute lectures were given.

5. The 2017/18 version of the course *Software Architecture and Design*, a compulsory for 1^{st} year MSc Software and Data Engineering students of Università della Svizzera italiana. The course was followed by 15 students. Across the fourteen course weeks, twelve 90-minute lectures were given.

Participation in the questionnaire was optional for students. We did not hold retrospective sessions for the Functional and Logical Programming courses in FIIT (described in subsection 7.1.1). An overview of courses with the number of

Table 3.2. Overview of retrospective sessions on ASQ and its question types. **Legend**. **#SQ** number of students that participated in the questionnaire. **#QT** number of question types that were evaluated. **QTN** name of the question types under evaluation.

Course	**#SQ**	**#QT**	**QTN**
Software Atelier III 2015/16	19	6	Multiple Choice, Text, CSS Select, Highlight, JS function, Classify
Web & Database Technology 2016/17	11	6	Multiple Choice, Text, Highlight, JS function, Classify, SQLlite
Web Atelier 2016/17	27	6	Multiple Choice, Text, CSS Select, Highlight, JS function, Classify
Web Atelier 2017/18	7	6	Multiple Choice, Text, CSS Select, Highlight, JS function, Classify
Software Architecture and Design 2017/18	13	5	Multiple Choice, Text, Highlight, JS function, Classify

students on each course that participated in the questionnaire and the number and name of question types that were evaluated is presented in Table 3.2.

3.4.2 Results

In general all question-types got favorable results as shown in Table 3.3. We should point out at this point that since participation in the questionnaires was optional, maybe only motivated students participated. This may have skewed the results towards the positive side. The best score was achieved by the *Multiple choice* type. Maybe it was due to its simplicity in usage and because recognition question require less retrieval effort than cued or free recall questions.

The live programming questions: *JS function*, *CSS Select* and *SQLite* also fared well with scores of 4.01, 4.04 and 4.07 respectively.

The *Highlight* question type got the lowest rating. This is partially explained by its implementation complexity. There were bugs that got solved over the years which is reflected in increasingly favorable ratings in Figure 3.4 which shows the temporal evolution of ratings over the courses. Another issue that we observed in the classroom and in the comments of the evaluation was that sometimes

students did not know how to use it.

In terms of textual feedback we observed that the responses focus mainly around general testimonials, bug reports and feature requests. In the first category we saw a lot of positive feedback, especially on the live-programming questions, like “Multichoice were great, although much much easier than the text input”(multiple choice); “Very easy to use.”(text); “Very fun.”, “The live feedback you get is great.”, “Nice sandbox tool”(JS function); “I think this was well implemented.”, “Straight-forward and intuitive - only the items might be a bit small”(classify); “It works nice, even though i do not like css” (for CSS Select); and “This is one of my favourite features. It is well executed and simulates a real SQL shell well.”(SQLite).

Regarding bug reports, the highlight question type had clear problems indicated by comments like “It was a bit buggy” and “It is a bit unintuitive - often I end up selecting the wrong thing. Also, multi-line selections often [illegible] wrong selection behaviors. I do like the heat[map visualization].”. We addressed a lot of the problems; this was reflected in the lack of bug reports in the last two sessions of the questionnaires (Web Atelier 2017/18 and Software Architecture and Design 2017/18). Another common bug report had to do with buggy behavior on small screen sizes (such as smart phones or tablets).

In terms of features requests we observed three trends: one towards explaining how a question type works, for example “Make it clear whether one or more answers can be selected” (multiple choice) and “At the beginning it was not clear how to use it, maybe some kind of tooltip for distracted people like me...” (highlight); the second towards providing hints and/or explanation of the correct answers, such as “maybe add hints” (text); and the last one towards moderating “bad” behavior like prohibit changing answers when the teacher gives feedback (“Is it possible for the presenter to close a question? I have seen multiple times that students changed their answer after the correct answer is displayed, updating the graph on the screen and distracting the students from the explanation the lecturer is giving.”) and censor (“It gets abused a lot, maybe give the presenter the option to show/hide answers”). Regarding the last trend we believe that is also has to do with the competition between students since they suggested changing the list of student submissions that some question types have with a full screen view of the submissions.

Table 3.3. Overview of retrospective sessions on ASQ and its question types. **Legend**. **MMC** Mean rating for *Multiple Choice*. **MT** Mean rating for *Text*. **MJSFB** Mean rating for *JS function*. **MCSS** Mean rating for *CSS Select*. **MHL** Mean rating for *highlight*. **MCL** Mean rating for *Classify*. **MSQL** Mean rating for *SQLite*. **#SQ** **#SQ** number of students that participated in the questionnaire.

Course	MMC	MT	MJSFB	MCSS	MHL	MCL	MSQL	#SQ
Software Atelier III (2015/16)	4.11	3.74	3.63	3.63	2.95	3.26	–	19
Software Architecture and Design (2017/18)	4.23	3.54	3.67	-	4.08	4.23	–	13
Web Atelier (2016/17)	4.30	3.96	4.19	4.23	3.50	3.91	–	27
Web Atelier (2017/18)	4.14	4.14	4.33	4.43	4.29	4.33	–	7
Web and Database Technology (2015/16)	4.55	4.45	4.50	–	2.90	4.17	4.64	11
Total	**4.26**	**3.92**	**4.01**	**4.04**	**3.45**	**3.85**	**4.07**	**77**

3.5 Summary

This section presented the main pillar of our work: the ASQ Web application for Interactive Web presentations. We explained the rationale behind the supported interactions, described the impact of introducing it in a classroom and delved into the question types and the rest of the interactive widgets we built to support active learning. We concluded with the results of quantitative and qualitative evaluation surveys for our question types obtained by students. In the next chapter, we will discuss the architecture of the tool and place a strong focus in its plugin system which is the basis for the implementation of its interactive widgets.

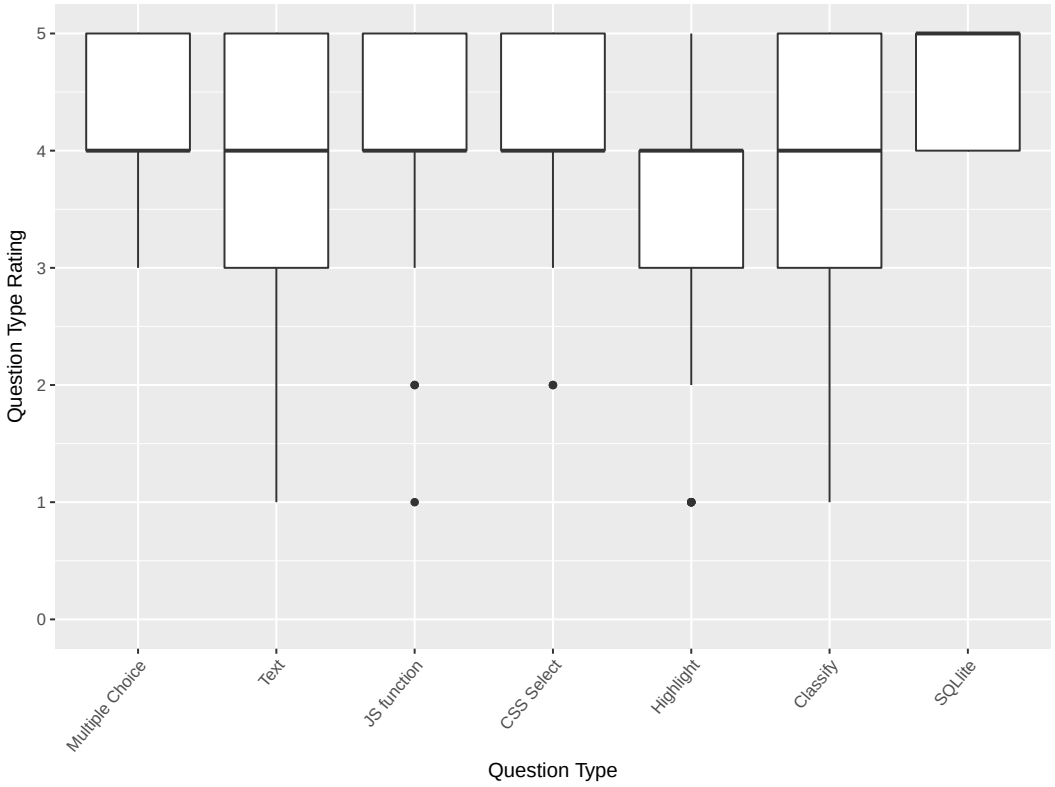


Figure 3.3. Boxplot of `ASQ` question type ratings.

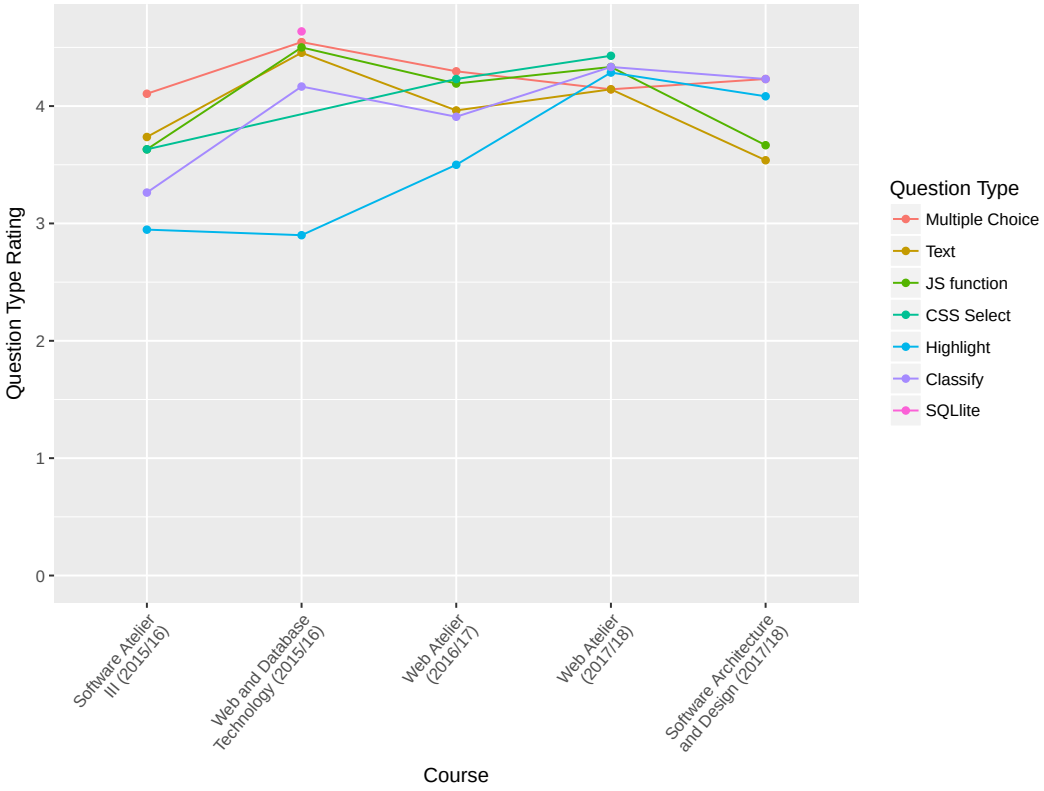


Figure 3.4. Temporal evolution of rating for the ASQ question types.

Chapter 4

ASQ Plugin System

4.1 Requirements

The design of ASQ was driven by the need to bring active learning and real-time classroom analytics in lectures with large audiences. While this chapter focuses on the plugin system of ASQ that is engineered to allow the creation of powerful question types, we think it is important to list the requirements for the whole system to better illustrate some of the design factors that influenced the implementation of the design system. The primary functional requirements for ASQ are:

Delivery of presentations that contain slides with interactive question types. ASQ delivers a unified experience of rich HTML5 presentations with embedded interactive question types to classroom audiences through a Web browser.

Real-time data collection from viewers and feedback from presenters to viewers. ASQ needs to collect in realtime quiz submissions from viewers and usage data from both viewers and presenters and persist them for later retrieval and analysis.

Automatic/Manual/Peer Assessment Instructors and teaching assistants should be able to manually grade submissions from all question types and especially those that are hard to automatically assess. To refrain from distractions during the *answer* cycle students should be able to assess submissions from their *peers* after they have submitted their own attempt.

Visualizations, Clustering and Analytics. To quickly give presenters the "big picture" of their audience submissions, most question types in ASQ are designed to use visualizations and clustering of results. Moreover, analytics regarding student behavior dynamics, such as attention levels help presenters with class orchestration.

Low latency bi-directional communication. ASQ needs to perform a lot of tasks with low latency requirements, such as slide synchronization between presenters and audience; collection of submissions; collection of questions from the audience to the presenter; and usage data collection.

Display audience submissions to presenters. Due to the limited classroom time audience submissions need to be gathered effortlessly. To control the cognitive load that is imposed on presenters when going through every submission, it is crucial to display submissions to the presenter in ways that facilitate giving them the "big picture" of the level of comprehension of the audience.

Important non-functional requirements include:

Scalability. ASQ needs to be able to support multiple live presentations with hundreds of users. ASQ is designed to horizontally scale content delivery and real-time communication with clients.

Extensibility. Our goal is to support various domains of knowledge in STEM fields that require specialized question types and presentation flows. ASQ is extensible via plugins that are built on top of the "asqium" plugin system that is explained in detail in section 4.2. Moreover, the backend business logic architecture of ASQ is modularized and layered to enable easy extensibility and configurability of the core features.

Resilience. When using ASQ to deliver presentations, slide delivery and the quiz lifecycle are instrumental to the teaching process. It is crucial that they operate as smoothly as possible even under heavy load, network interruption and various other disaster scenarios.

Flexible User Privileges and Privacy. Another requirement is to enable interactions between participants that can benefit their learning(e.g., peer assessment

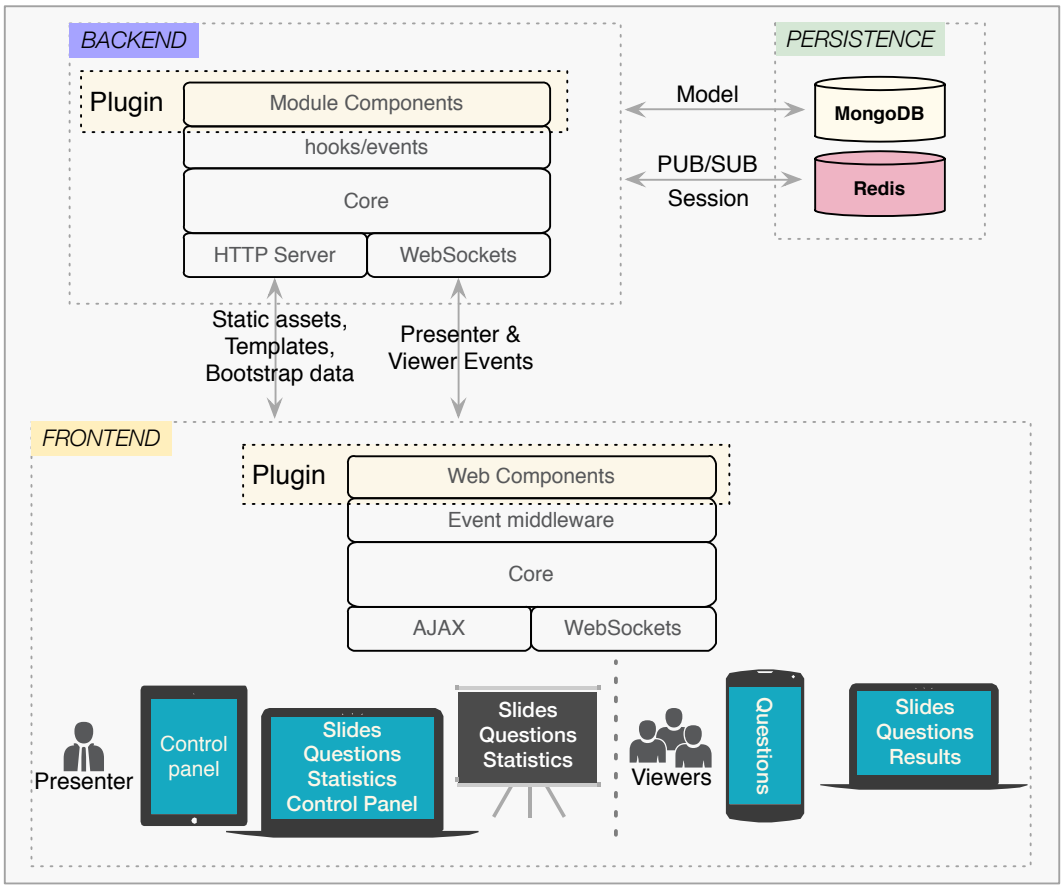


Figure 4.1. ASQ architecture with the plugin system

or comparative evaluations) while respecting their privacy to encourage participation in a safe environment. For example, some presenters may choose to have their audience members remain anonymous. Or for some other presentations, a student may become the presenter and the teacher a viewer. To this end we engineered an Access Control List (ACL) that affords users different privileges and privacy options per presentation.

As mentioned before, while this list of requirements was taken into consideration during the design of the ASQ as a whole, in this chapter we will just focus on the plugin system of ASQ which we consider the most important contribution in designing systems with similar requirements as ours. To provide the reader with the bigger picture Figure 4.1 displays the overall architecture of the application which is discussed in more detail in subsection B.1.1

4.2 The Asqium Plugin System

Questions are an integral part of ASQ, and one of the early design decisions was to allow content authors to create custom question types that can fit into the existing flow and extend it with new functionality, hence the need for a powerful plugin system to make the platform versatile and extensible:

- Content authors should be able to create custom question types or extend existing ones. Types can range from simple multiple choice questions to advanced code editing questions with automatic unit-test assessment.
- Content authors should be able to create custom feedback logic and visualizations to target different presentational needs and accommodate for heterogeneous data coming from different question types. To better illustrate this, let us assume two different question types: multiple choice questions, where the answer is the combination of the checked options; and a highlight question, where the answer is the highlighted parts of a given text. Whereas it makes sense to render audience responses as a barchart or a pie chart in the former case it may not provide a meaningful visual representation for a heatmap question. A better choice may be to create a heatmap which maps each character position of the text with a color intensity that is proportional to the number of times the specific character position was highlighted by the audience.
- Presenters should be able to enable lecture flow and interaction patterns that match the usage context, their teaching style, the applied pedagogy and the nature of the question types. Examples include: the ability to have synchronized slides between presenter and audience within the classroom and free navigation for viewers during studying (presentation context); the ability to display assessment results in real time versus a specific point in time, or individual versus aggregated results (teaching style); the ability to allow students to work individually or in groups (applied pedagogy); the ability to have automatic, self or peer assessment strategies (applied pedagogy and question type complexity).
- Easily swap data-mining and analytics plugins to compare different techniques and algorithms to increase teacher awareness.
- Easy integration of complex external services and data sources without polluting the system architecture.

Asqium, the ASQ plugin system, has been implemented as the foundation of ASQ. Although integral to ASQ, Asqium can be utilized in third-party software that has similar requirements as a library. The code for the plugin system implementation is available at `https://github.com/ASQ-USI/ASQ/tree/master/lib/plugin`. The back-end plugin base is an npm package which can be found at `https://github.com/ASQ-USI/asq-plugin`.

4.2.1 Design Goals

From our experience with the design of ASQ and after analyzing the architecture of several modern Web applications featuring real-time updates, we collected the following required characteristics that should be satisfied by a plugin system to enhance the extensibility of the Web application.

Application Domain Compliance The domain of each application dictates the respective entities and their privileges, the flow of information between them and the security constraints that govern them. Plugins should adhere to these constraints rules. As an example, in the case of ASQ, the presenter is in control of information flow which follows variations of the *ask-answer-assess-feedback* cycle. A plugin that would automatically show assessment results without the presenter's consent would violate the domain rules.

Open Event Model. Since plugins extend the application with new functionality, it is likely that they may introduce new events which may not be part of the existing 'information flow' events of the application. While the latter are expected to be gracefully handled by a plugin, it is also very important to be able to extend the possible events exchanged within an application with custom ones, as long as they do not violate the main application flow.

Persistence Flexibility. Web applications often use more than one storage technology to tailor the way different parts of their data model are managed. For example, some aggregation operation is performed on some data and the extracted result gets propagated to the clients in real-time; similarly we may also store the raw data for deferred processing. These could result in using a simple fast in-memory key-value storage and slower document-based disk storage respectively. Plugins should be able to take advantage of both strategies.

Encapsulation and Theming. While encapsulation seems like an obvious desirable characteristic which is readily available by most modern programming languages and Web frameworks, until recently it was very hard to create encapsulated front-end components due to limitations of the HTML/CSS platform. Even if the component authors are careful enough to use high specificity CSS selectors there is no guarantee that the rest of the third-party style rules present in a page will not target the widget markup. To avoid bleeding JavaScript global variables [kan09], developers may choose to use closures. But this approach will make it harder to expose functionality of their widget to third-party code. Some of these problems have workarounds that fail to conceal the fact that JavaScript was not designed for large scale applications by imposing limitations. For example the CommonJS and AMD standards allow better encapsulation of JavaScript code at the cost of precompilation which negates the role of JavaScript as an

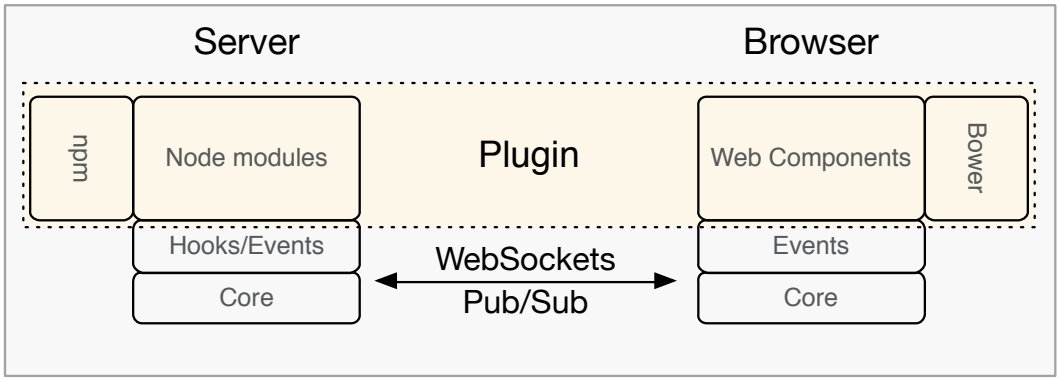


Figure 4.2. Plugin structure: Back-end modules and Front-end Web components

interpreted language. Theming could make matters worse, since rules should either be very specific, which leads to hard to maintain codebases; or generic which could result in unintentional style rule leaks towards non-target elements.

Isomorphism. Developing isomorphic [BÍ15] Web applications has the benefits of using the same code both on the front-end and the back-end, a milder learning curve for new developers, better communication between front-end and back-end teams and smaller technology stack. JavaScript is the de-facto browser language and demonstrates good asynchronous performance server-side which renders it an ideal candidate for isomorphic applications.

Easy Deployment and Publication As any piece of software, plugins may go through many iterations and releases. Common steps in these cycles are testing the code in isolation, then functional testing within the host system. If everything is complete in terms of target features and successful testing, the plugin gets released so that its users can update to the latest version. This process can be tiresome and error prone since there are many file transfers involved and a lot points of failure: moving code between the plugin's source code directory, which in most cases is under some kind of revision control, and the target host system; deploying to remote servers; separating the client-side from the server-side components; and ensuring the compatibility of different versions in plugin-to-plugin dependencies. Streamlining these processes can help both plugin developers and consumers.

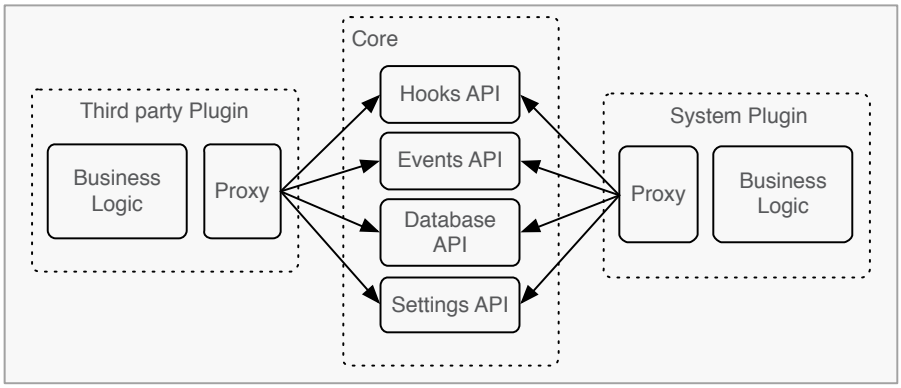


Figure 4.3. Proxy object used as the entry point façade for the plugin contributed business logic to access the core APIs

4.2.2 Server-side Plugins

Server plugins are implemented as npm modules. The only mandatory dependency from the plugin system is extending a base class which offers some conveniences for the developers like declarative mapping of hook names to callbacks and lifecycle methods. There are four lifecycle methods: `install`, `uninstall`, `activate` and `deactivate`, which are called from core when a system user tries to perform one of the corresponding actions. This allows plugins to perform tasks like creating settings, populating persistance store or performing cleanup.

Proxy object Proxy objects are used as a single façade interface between plugin modules and the host system. A proxy object (Fig. 4.3) exposes all the available APIs that a plugin may use to interface with the core, like hooks, events, settings and database APIs. The implementation of a proxy object is provided by the core. Each time the core instantiates a plugin, it passes the plugin constructor a new instance of a proxy object. Conversely, the core does not directly call methods on plugins: instead it executes hooks or publishes events to which the plugin can subscribe. This ensures that the core remains decoupled from the plugins.

4.2.3 Front-end Components

Web Components is an umbrella term covering four Web Technologies: Custom elements [57], Shadow DOM [59], the HTML template element and HTML imports [58]. The synthesis of these technologies allows the creation of HTML elements that have encapsulated CSS styles, their own DOM tree (Shadow Dom) and are also JavaScript objects which helps mitigate the global variables bleeding

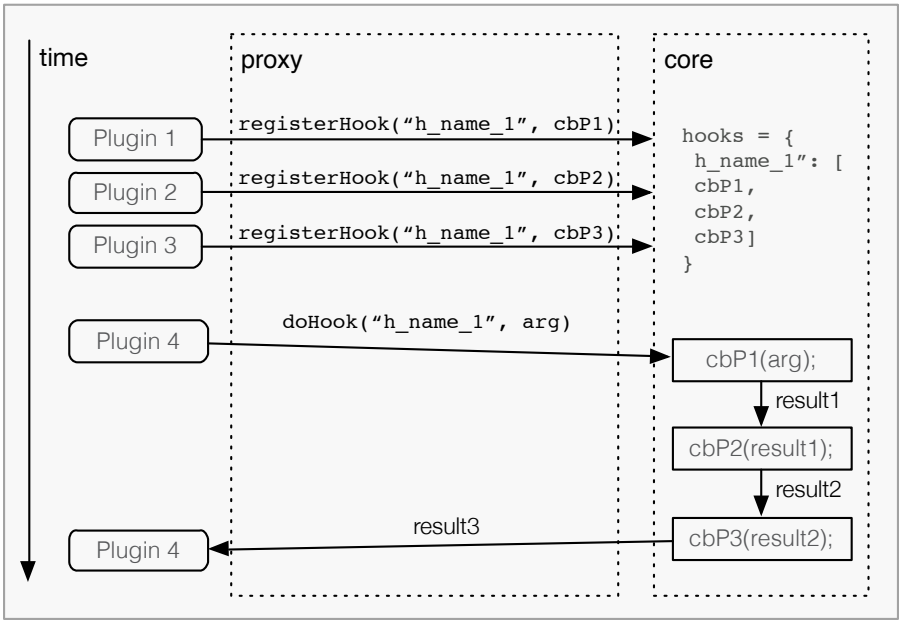


Figure 4.4. Hook lifecycle: registration and chained invocation

problem which affects many Web applications. Custom elements can be accessed and manipulated with regular DOM methods since they reside and are part of the DOM [136]. Shadow DOM allows us to separate markup that describes content from markup that is purely presentational. Implementation details can be hidden from the user which results to components with succinct markup [180].

We use the Polymer library [Pol18] which builds on top of the Web Components technology. Polymer polyfills [Sha10] missing Web Components Technologies in case they are not available in a browser and adds some functionalities like data-binding and declarative element registration. A typical Web Component defines one or more Custom Elements that encapsulate the User Interface and front-end business logic of the plugin. The plugin subscribes and publishes events to communicate with the rest of the application. Thus, the only dependency between front-end components and the front-end core of the application is the pub/sub implementation which is already present in the form of the EventTarget interface that most DOM Elements implement.

4.2.4 Communication and Message passing

Server-side We distinguish two modes of server-side task execution and discuss the corresponding communication patterns between participating plugins to ac-

complish them:

1. **Input transformation tasks with completion acknowledgement** These are tasks where each participating plugin applies a transformation to some input. The result is passed to the next participating plugin until there is no plugin left in which case the final output is returned to the initiator of the task. Here we only consider the case where order is not important. Participating components can process these tasks in parallel or asynchronously. This is because all participating components are operating on the same data which can lead to unexpected results. Nevertheless, a plugin may execute asynchronous code, which is encouraged for I/O operations, as long as it returns control to the callee when it has finished execution in order to proceed with the next plugin. This can be implemented with callbacks and/or Promises [67]. Examples of such tasks are: knowing when an answer from a student has been processed and persisted to the database in order to update progress information; composing a piece of information that needs to be sent as part of a single acrshorthttp response to the client, like the head of an HTML document.

To target this type of execution, we implemented a hook system. The hook system allows plugins to register for a hook providing the name of the hook and the function (callback) to be invoked when this hook is executed. Hooks identify specific tasks of the Application that require sequential execution and result passing between chained invocations of logic that is contributed by one or more plugins. Hook callbacks have an arity of one, with the only argument being the result of the previous callback execution for the same hook. The initial value for the argument is provided from the initiator of the hook execution similar to a reduce function. Any plugin can initiate a hook execution and any plugin can register. The `doHook` function in Listing 4.1 executes all callbacks for a specific hook. Notice the use of `Promise.reduce` that waits for each task to return either a Promise or a value and then continues with the next callback.

```
1  function doHook(name){
2
3     if(! this.hookCbs[name]) return Promise.resolve(true);
4     var args = Array.prototype.slice.call(arguments, 1);
5
6     //execute callbacks sequentially
7     return Promise.reduce(this.hookCbs[name], function(arg, hookFn){
8       return Promise.resolve(hookFn(arg));
9     }, args);
10 }
```

Listing 4.1. Executing a hook by triggering contributed callbacks

2. **Decoupled asynchronous tasks** These are tasks that are executed as response to some significant change in the state of the system (event). The initiator of the state change has no knowledge of the other components that are interested in the change. This approach has the benefits of loosely coupled components and asynchronous executions of tasks (which can boost performance). Examples of such system changes that components may want to subscribe to are: 'a new user has come online' or 'a checkbox was ticked'.
Such problems can be modeled using an event-based approach: Event notifications are produced and propagated with message passing according to the pub-/sub pattern. A plugin subscribes for an event which may be triggered from the core or other plugins. The dispatcher of the event has no expectations for a completion acknowledgement or some result to be returned. Event-based communication is encouraged in JavaScript development since there's native support in both the server (EventEmitter in Node.js) and the client (native DOM support with EventTarget).

Client-side Our approach is to use *publish/subscribe* style communication for plugins and the core using an EventEmitter-like library, and within the plugins a combination of DOM events communication and method invocation. The reason we do not use EventTarget DOM elements as event producers and consumers for non-DOM-related events is to maintain a uniform (isomorphic) interface for event creation and handling across both modules and Web Components.

The host application awaits for all plugins to be instantiated by listening for a `WebComponentsReady` event. Then it dispatches through the `document` an `app-ready` event for which plugins should have a listener for. As a payload to the event message is the EventEmitter instance that is going to be used as the event bus of the whole application. Plugins can subsequently publish/subscribe to events on the EventEmitter instance. The `app-ready` event is the only document dispatched event between a plugin and the application. The rest of the communication is carried out through the EventEmitter.

```
1 var eventBus = new EventEmitter2();
2 document.addEventListener("WebComponentsReady", function(){
3   var event = new CustomEvent('app-ready', { 'detail': {appEventBus :
        eventBus} });
4   document.dispatchEvent(event);
5 });
```

Listing 4.2. Seeding the EventEmitter instance to front-end plugins

```
1 document.addEventListener('app-ready', function(evt){
2    evt.detail.appEventBus.on('asq:question_type',
```

```
        this.onQuestionType.bind(this));
}.bind(this));
```

Listing 4.3. A front-end plugin receiving the EventEmitter instance and using it to subscribe for events

Server-client plugin communication Real-time Web applications establish low-latency, low-overhead bidirectional communication streams between client and server through WebSockets. Plugins from both sides register for events on an intermediate layer that receives events from the WebSockets layer. The intermediate layer is tasked with filtering events and only re-emitting those that can be consumed by plugins. An EventEmitter-like instance is used to publish or subscribe to events. To send custom messages to the server, Web components use dedicated events, specifying their unique plugin name in the 'type' field of the event. This ensures that modules will process this event since they can subscribe to receive it.

Server generated events targeted to client counterparts of a plugin use the same event structure but an extended identification mechanism. The rationale behind this is that there can be many connected clients that have an instance of the target Web Component and that may be in the scope of a specific event. For example a user may have opened two instances of our application in two separate browser windows; or we may want to target all users that have a specific role in our application, e.g. all the administrators. Our system uses the Socket.IO library to provide WebSocket functionality which allows grouping socket connections in rooms and namespaces [Rau14]. This allows a server plugin to easily target logically grouped clients. Listing 4.9 demonstrates a simple use case of this pattern where a server-side plugin emits an event which targets its client-side counterpart running on clients that belong to the `ctrl` event namespace.

4.2.5 Persistence

The persistence APIs and behaviour is designed under the assumption that the persistence layer comprises schema-less document or key-value stores like MongoDB and Redis. Enforcing the schema of the data is possible, and recommended, at the business logic layer of the system.

API interface Proxy objects provide plugins with an interface to the data stores used by the core. The interface is intended to help to accelerate common development tasks for plugins and reduce common pitfalls, as opposed to providing real security.

In our experience, a plugin usually needs two types of persistence data. The

first is general settings for the plugins' behaviour, for example: activated vs deactivated states, options of the control panel of the plugin, general configuration options and more. RDBMS based applications offer one or more tables dedicated to this cause. RDBMS require data stored in tables to conform to strict schemas. A common strategy to enable storage of arbitrary data in a single table, is to serialize all data types, including hashmaps and arrays, into strings. Such a concession is not required in document based databases, therefore all types of setting data from all plugins and the core can be stored in one collection (the equivalent of a table).[1]

The second type is data for the plugin implemented business logic. In this case plugin data may either share the same structure as the core generated data or introduce new structure. Again, small mismatches in similar data can be mitigated, as we will demonstrate in the evaluation section, by the schema-less nature of document based stores; allowing the plugins to share the same collections as the core. There are cases however, where plugins need to store data that are little to not at all related with the existing data. In such cases plugins can create their own collections.

Schema and Data migration Plugins for traditional RDBMS usually have some logic that migrates the schema and stored data of related database tables from one version of the plugin to another. A lot of modern document based or key-value stores are schema-less and as such there is no need for schema migration. However data should be kept consistent in structure, so a data migration plan is necessary. These operations can be performed during the `install` lifecycle method of a plugin.

4.2.6 Security and Plugin Isolation

Role-based Access Control (RBAC) is a popular alternative for authorized access to system resources. Roles are exposed to plugins either via the Settings API or with direct access to the persistence stores.

At runtime, a plugin's code should run in isolation and in case of failure, if possible, it should not cause the rest of the application to fail as well. To minimize dependencies and thus the possibility for failures the only way a plugin can interface with the core is through the proxy object instance. The event and hooks systems are agnostic of the presence of plugins. All lifecycle methods of plugin modules that are called from the core are contained in try-catch blocks.

[1] In some cases performance may be affected if some conditions are not met. For example collections with documents that vary greatly in size may induce write penalties.

This holds true for hooks as well: when a hook is executed the code that initiates the hook execution should catch and handle any errors that may occur from invoking a registered callback that is a plugin method.

4.2.7 Deployment and Release Engineering

Npm is the package manager for Node.js modules. It is configured via a `package.json` file. Bower is a package manager for front-end assets like JavaScript libraries and CSS frameworks. It is configured by a `bower.json` file. Our plugin framework uses both to specify the modules and Web components contributed by the plugin respectively (see Fig. 4.2). The host system can specify the plugin as a dependency using the respective package managers. This design has a number of benefits:

1. reuses established package managers.
2. allows for a single code repository for the entire plugin.
3. enables to ignore files that are not required by a specific counterpart of the plugin. Npm uses either the `files` field in the `package.json` file for an inclusive list of files or a `.npmignore` file in the project's root directory that lists files to be excluded. Bower supports an `ignore` field in the `bower.json` file to specify which files to exclude.
4. makes it convenient to use the latest version of the plugin code while testing. This can be achieved either by using the tip of a specific code branch or by symlinking the package (be it an npm or a bower package) using built-in package manager commands.

The npm module part of the plugin can be deployed by the command `npm install <package-name>` which will install it in the `node_modules` directory of the server-side code. Given correct implementation of the server-side plugin lifecycle methods, plugins can be installed, uninstalled, activated and deactivated while the application is executing, allowing us to hot-swap implementations while testing. The Web Component part of the plugin can be installed by specifying it as a dependency in the `bower.json` file or the front-end components of the host system. Additional building steps can be implemented to allow code transformation, minification and other release engineering tasks.

4.2.8 Crafting a plugin for **ASQ**

In the remainder of this section we will demonstrate how we can use the presented plugin system to craft a question type plugin [244] for a highlight question. We will name the plugin 'asq-highlight'.

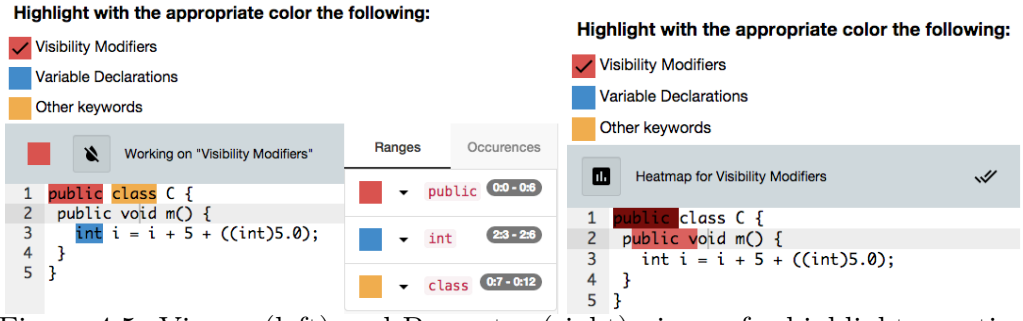


Figure 4.5. Viewer (left) and Presenter (right) views of a highlight question

Creating the User Interface A custom question type has different appearance and functionality based on the role and view of the user. In our case, an audience viewer faces a text editor with highlight capabilities showing the text to be highlighted (left of Figure 4.5). On the beamer view (right of Figure 4.5) the presenter can toggle between two modes: a heatmap over all audience answers or the correct solution. On the control panel, the presenter can display the highlight solution for any of the audience members or compare two or more answers with a heatmap.

To implement the User Interface we create a Custom Element, `<asq-highlight>`, that will be the façade for two role-based elements `<asq-highlight-viewer>` and `<asq-highlight-presenter>`; Listing 4.4 shows this in effect. The template of `<asq-highlight>` uses a conditional template (a Polymer feature) to render the appropriate element based on the role attribute of `<asq-highlight>`. Then it forwards related attributes and the content (also known as distributed nodes) to the role-based elements.

Content authors can embed an `<asq-highlight>` element in their presentation using code similar to Listing 4.5. Notice the definition of two more elements, `<asq-stem>` and `<asq-hl-color-task>`. We can define more than one elements that can be distributed with a Web Component to create structural elements that encapsulate presentation resulting in cleaner and shorter markup.

```
1  <template if="{{role == roles.VIEWER}}">
2    <asq-highlight-viewer mode="{{mode}}" theme="{{theme}}"
         fontSize="{{fontSize}}">
3      <content></content>
4    </asq-highlight-viewer>
5  </template>
6  <template if="{{role == roles.PRESENTER}}">
7    <asq-highlight-presenter mode="{{mode}}" theme="{{theme}}"
         fontSize="{{fontSize}}">
8      <content></content>
```

```
  </asq-highlight-presenter>
</template>
```

Listing 4.4. Rendering elements based on roles (using the Polymer library). Template of the `<asq-highglight>` element

Implementing Server-side business logic The server-side module of our asq-highlight will respond to two hooks: `parse_html` which is triggered when a user uploads an HTML presentation file; and `answer_submission`, triggered when an audience member submits an answer to a question.

For `parse_html` we are interested in extracting the representation of all `<asq-highlight>` elements present in an HTML string and persisting them in the 'questions' collection of our models persistence store (MongoDB). Listing 4.6 shows the implementation of the callback for `parse_html`. `this.asq` refers to the proxy instance that in this particular case allows to interact with the persistance store to persist the extracted questions.

```
<asq-highlight theme="textmate" mode="java" fontSize="1em">
   <asq-stem><h3>Highlight with the appropriate color the
       following:</h3></asq-stem>
   <asq-hl-color-task color="d9534f">Visibility Modifiers</asq-hl-color-task>
   <asq-hl-color-task color="428bca">Variable Declarations</asq-hl-color-task>
   <asq-hl-color-task color="f0ad4e">Other keywords</asq-hl-color-task>
   <code>public class C {
  public void m() {
    int i = i + 5 + ((int)5.0) + ((int)5f);
  }
}</code>
</asq-highlight>
```

Listing 4.5. markup to create the showcased highlight question

```
function parseHtml(html){
  //cheerio is a Node.js library for \acrshort{html} manipulation
  var $ = cheerio.load(html, {decodeEntities: false});
  var hlQuestions = [];

  $(this.tagName).each(function(idx, el){
    hlQuestions.push(this.processEl($, el));
  }.bind(this));

  //return Promise that resolves with the (maybe modified) html
  return this.asq.db.model("Question").create(hlQuestions)
  .then(function(){
    return Promise.resolve($.root().html());
```

```
14   });
15 }
```

Listing 4.6. `parse-html` hook callback for asq-highlight

When a viewer submits an answer to a set of questions, a core plugin fires the `exercise_submission` hook and for each individual question an `answer_submission` hook (Listing 4.7). Asq-highlight provides a callback for `answer_submission` in order to persist the Answer to the database as in Listing 4.8.

Notice the invocation of `this.calculateProgress` which calculates how many of the audience members have answered the question identified by `questionUid` and brodcasts the result to all connected clients. This method executes asynchronously and is a good example of event notification. The abbreviated version in Listing 4.9 shows how the event published from the server can target multiple clients.

```
1  try {
2    sanitizeSubmission();
3
4    //execute ‘exercise_submission‘ hook
5    yield hooks.doHook("exercise_submission", submission)
6
7    yield Promise.map(submission.answers, function(answer){
8      //execute ‘answer_submission‘ hook
9      hooks.doHook("answer_submission", answer);
10   });
11
12 } catch(err) {
13   //if a plugin fails to execute the hook we will catch the error here
14   logger.error(err.message, { err: err.stack });
15 }
```

Listing 4.7. triggering submission hooks in plugins

```
1  answerSubmission: coroutine(function *answerSubmissionGen (answer){
2    var questionUid = answer.questionUid
3    this.validateAnswer(answer);
4    yield this.asq.db.model("Answer").create({
5      question : questionUid,
6      answeree : answer.answeree,
7      session  : answer.session,
8      submitDate : Date.now(),
9      submission : answer.submission,
10   });
11   this.calculateProgress(answer.session, ObjectId(questionUid));
```

```
12   //this will be the argument to the next hook
13   return answer;
14 }
```

Listing 4.8. submission hook execution

```
1  var event = {
2   questionType: 'asq-highlight',
3   type: 'progress',
4   questionUid: question_id.toString(),
5   heatmapData: JSON.stringify(heatmapData),
6  }
7  this.asq.sendSocketEventToNamespaces('asq:question_type', event,
       session_id.toString(), 'ctrl')
```

Listing 4.9. sending an event to all presenter clients

4.3 Summary

In this chapter we presented the overall architecture of ASQ. We listed the requirements behind its design, its data model and presented core functionality. We then moved to the topic of plugin design for widgets that span both the front-end and the back-end of Web architectures and which address the extensibility requirement for implementing powerful question types. We shared extended implementation insights and presented an example question type implementation.

With this chapter the presentation of the Web engineering work of this dissertation comes to an end. In the next one we will move to educational landscape and present findings arising from introducing ASQ in real-world classrooms which demonstrate ASQ's potential as an instructional tool in the modern classroom.

Chapter 5

ASQ as an Educational Study Platform

From the outset ASQ was designed with features that make it suitable to conduct research on technology-enhanced learning in real classrooms. ASQ's fine-grained logging abilities allow us to track second by second to what extent students are engaging with ASQ which in turn enables insights into student behaviour dynamics. This setup enables us to conduct "in situ" experiments.

To validate our design, in this chapter we present results from three experiments that occured during two deployments of ASQ in real-world courses. The application was used for delivering the lectures of two consecutive versions (2015-2016, 2016-2017) of the *Web and Database Technology* course in the Delft University of Technology. In the first experiment we examine if and how suitable is ASQ to assess the attention and learning outcomes of students by analyzing recorded student-generated events during the students' interaction with the application. Next, we look into the usability aspects of the tool as subjectively perceived by the students that used it. We calculated System Usability Scale (SUS) scores from the majority of the sessions and we present an analysis of the results. Finally, we are interested in the cognitive load that the platform inflicts upon the instructors. To gather a subjective estimation we asked our subjects to fill in a NASA Task Load Index (NASA-TLX) questionnaire after each lecture and we analyzed the results for which give a brief report.

5.1 Case Study: Measuring Student Behaviour Dynamics in a Large Interactive Classroom Setting

Encouraging positive use of digital technologies in the classroom is not new. Previous work addressed the effect of personal response systems (colloquially known

as "clickers") on student attention and engagement [119; 81]. Other studies have explored the impact of laptops in the classroom for note-taking [188; 155] or undirected use [93]. While providing insights on the relationship between digital technologies and students behaviour, many prior works suffer from one main technical limitation: they assume the students' devices to be complementary to, and not integrated with, the lecture experience. In our work, we take the next step and turn the students' devices from potential distractors into a teaching and communication tool by *seamlessly* interleaving lecture material and *complex* questions such as programming questions in each student's browser.

Previous work is also largely affected by a methodological limitation: experiments took place in a controlled setting, where the students' identities were known, and they were explicitly assigned to experimental conditions that could have harmed or influenced their learning experience.

These issues of privacy and fairness are not compatible with the requirements of a real-world course, where students must be guaranteed equal treatment, and privacy must be preserved. These requirements are at the center of our use cases, and defined our experimental methodology: participation was optional, students' identity concealed, and the learning set-up equal for all participants. Intuitively, changing the experimental conditions might make our experiments not comparable with previous work. To this end, we first focused on the following research question:

5.1.1 Threats to Experimental Validity

In our experiments presented in this dissertation, due to the requirements of privacy and fairness, we opted for a design where one student population received a treatment (the use `ASQ`) in multiple sessions. After each session we measured the effects. This design, while fair to our students, has no pretest and no control group which makes it open to the following threats of internal and external validity [37]. *History*, or what happens between the events, is a consideration, especially since the experiments lasted for weeks. *Maturation*, the biological and psychological processes which systematically vary with the passage of time, is also relevant due the long duration of the experiments. Moreover, since participation was optional, *mortality* (differential drop-out of persons from the group) may explain some of the results. The three sources of invalidity we mentioned so far concern the internal validity of the experiment. *Interaction of Selection and `ASQ`* is an external source of invalidity in which the student populations of the experiments may not be representative of all the learning settings in which we wish to deploy `ASQ`.

RQ1.1 *To what extent can a Web-based, privacy-preserving teaching tool be reliably used to assess student behavior dynamics and learning outcomes of students from their interactions with the tool?*

To answer this question, we analyse the ASQ user logs collected throughout our target courses, and compare the obtained results with the findings collected from a systematic analysis of the literature. For instance, it is well-known that the activity of answering practice questions is benefiting students by increasing their engagement in the classroom [149]. Confirming these results, provides evidence of the suitability of ASQ as a platform for longitudinal "in-situ" experiments, thus enabling the investigation of additional research questions.

5.1.2 Methodology & Use Case

To answer our research question, we will first discuss the privacy policy of ASQ when collecting student generated data. Then we need to define engagement metrics based on the events emitted by each student's browser (section 5.1.2). Finally, in section 5.1.2 we introduce our target undergraduate course in more detail.

User state and privacy

ASQ does not require students to log in, and events are captured as soon as the browser connects to a running ASQ presentation session. Closing the browser tab that renders the ASQ presentation will disconnect the student. Each student is given a unique identification token for each presentation session and active connection. The token expires when the browser session expires, i.e. when the Web browser that established the initial connection to ASQ closes. This allows us to associate students with the events they generate, while preserving students' privacy across multiple lectures. ASQ tracks different types of events (Table B.1 and Table B.2) that are generated by the browser during a presentation session; occurrences of such events are immediately sent to the ASQ server for logging. Examples of events include: (i) connecting to the ASQ presentation; (ii) submitting an answer to a question; (iii) switching to another browser tab, or to another application, which may make the ASQ window invisible.

Figure 5.1 provides an example of the sequence of events emitted by three students during the first 15 minutes of a lecture. The first student has a low level of engagement as he immediately hides the ASQ window and then disconnects after 3 minutes without answering any questions. The second student shows

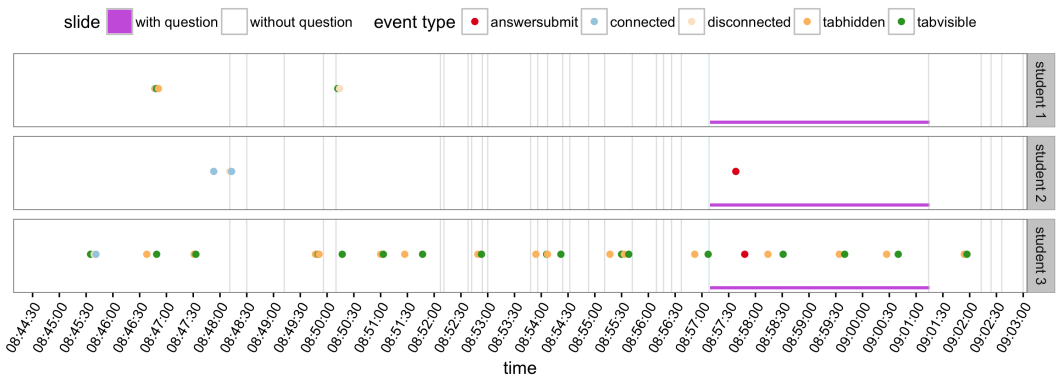


Figure 5.1. Sequence of events for three example viewers during the first 15 minutes of Lecture 2, HTML. Each vertical line represents a slide change — there are 29 slides shown in total, one of which is a question slide.

a high level of engagement with the slides (which are never hidden after the initial connection) and also submits one answer to a question. The third student presents several context switches where ASQ is repeatedly hidden and shortly thereafter becomes visible again. This behavior continues also after the student submits an answer.

Modeling slide and question engagement

For each student, events are aggregated in order to compute *slide* and *question* engagement metrics. The former refers to engagement during the non-interactive parts of the presentation session, i.e. the content slides presented by the lecturer. The latter refers to engagement during the interactive (question-containing) slides of the session. We consider a student engaged with the slides if they are visible to the student in his or her browser. Conversely, we consider students not to be engaged if the lecture slides are not visible on their own devices (e.g. due to the devices being switched off, or used for other activities such as browsing the Web, chatting and so on).

We use the `tabvisible` and `tabhidden` events to detect whether the ASQ Web browser tab of the student is visible or not as well as the `answersubmit` event to detect if a student has submitted an answer to a question. We assume that a session s (full lecture from the time a presentation starts and ends in ASQ) of length $T(s)$ starts at second 1 and ends at second $T(s)$. For every student v, for every second t of session s we create an indicator variable $visible(v,s,t)$ which is

1 if the ASQ tab is visible and 0 otherwise. The mean slide engagement $MSE(v,s)$ of a student across s is the number of seconds the ASQ tab is visible, normalised by the session length:

$$MSE(v,s) = \frac{1}{T(s)} \sum_{t=1}^{T(s)} visible(v,s,t) \tag{5.1}$$

A student v's question engagement is exclusively defined over the interactive (question) slides in s; it is the fraction of questions $q \in Q(s)$ (with $Q(s)$ being the set of all questions in session s) that v submitted an answer for:

$$\text{MQE}(v,s) = \frac{1}{|Q(s)|} \sum_{q \in Q(s)} submitted(v,q) \tag{5.2}$$

Here, $submitted(v,q)$ is 1 if v submitted at least one answer to q and 0 otherwise.

Table 5.1. Lecture overview, reported in temporal order of delivery. **Legend**. #Q: number of questions in the slides. **#STU**: number of students in the classroom. **%ASQ**: the percentage of students in the classroom connected to ASQ. $\rho(S,Q)$: Spearman rank order correlation between slide and question engagement scores.

Lecture	**#Q**	**#STU**	**%ASQ**	$\rho(S,Q)$
HTTP	10	319	99.7%	0.72†
HTML	8	238	94.5%	0.58†
JavaScript	10	192	91.1%	0.69†
Database introduction	7	204	83.8%	0.60†
node.js	8	208	64.9%	0.66†
CSS	7	163	82.8%	0.72†
SQL introduction	9	196	75.0%	0.57†
SQL continued	8	157	87.9%	0.44†
Cookies & session	9	133	97.0%	0.51†
Advanced SQL	10	89	87.6%	0.64†
Web security	8	151	70.9%	0.50†
Conceptual design	13	83	65.1%	0.42†
ER logical design	8	125	68.0%	0.40†
NoSQL	8	89	76.4%	0.39†

Course overview

The data for this study was collected during the 2016/17 edition of a first year Bachelor course teaching Web technology and database concepts to Computer Science students at the Delft University of Technology. The course took place between November 14, 2016 and January 20, 2017. Table 5.1 presents the topic overview of the 14 lectures[1]; each lecture lasted 90 minutes, after 45 minutes a fifteen minute break occurred. The course topics were split across two instructors (I_1 and I_2). Every lecture contained between seven and 13 question of various types: multiple-choice, multiple-answer, highlighting, SQL-query creation and fill-in-the-blank questions;

5.1.3 Results

In this section we report on the analysis of the usage logs created by ASQ during the 10 weeks and 14 logged lectures of the course. First, in section 5.1.3 we analyze the *slide* and *question* engagement of students across lectures, and compare the findings with previous work to assess the suitability of ASQ as a privacy-preserving platform for "in-situ" experiments. Then section 5.1.3 we reflect on the lessons learned from using ASQ in the classroom.

RQ1.1: **ASQ** as an experimental platform

Table 5.1 reports the course attendance statistics, and data about the percentage of students using ASQ to visualise slides and to interact with questions. In each lecture, the students were counted ten minutes after the official start of the lecture.

Engagement across lectures. Over the course of the class, student attendance dropped from initially 319 to approximately 100 students — an attendance drop also seen in comparable courses at the same institution.

The usage of ASQ also fluctuated: while in the first three lectures more than 90% of the students used ASQ at least for some time during the lecture, novelty wore off and in later lectures the usage varied with between 65% and 80% of students connected to ASQ. Of those students that used ASQ, between 63% and 85% answered at least one question, while at most 27% of students submitted

[1] In total, the course contained 15 lectures; one had to be excluded from our analysis due to a faulty logging mechanism.

answers to all of a lecture's questions; in three lectures (all on database topics) fewer than 3% of students were such all-answer-submitting students. Across the 14 lectures we only observe 4 lectures (Conceptual design, CSS, HTML and HTTP) where at least half of the ASQ connected students submitted answers to more than half of the posed questions. These results are lower than expected, but not completely surprising. Not all students are equally compelled to experience lectures through their own devices (some students prefer pen and paper when being given the choice [5]), and their preference could vary according to topic. Also, questions varied in complexity; for topics like *SQL* (where SQL queries needed to be produced by students during the lecture), answering all questions was objectively harder than for lectures with mostly multiple-choice or multiple-answer questions. These results show that ASQ has been well-received by the course's students, and that a significant number of such students has been actively engaged with the lecture material and with questions.

Engagement within lectures. The fine-grained logging abilities of ASQ allow us to also investigate the students' engagement with ASQ beyond submitting responses to questions. Let us consider the students' *slide* and *question* engagement, as defined in section 5.1.2.

The distribution of students engaged with slides for each lecture is plotted in Figure 5.2. Focusing on the first lecture (HTTP), we observe that half of the students are engaged with the slides between 25% and 76% of the lecture time. In subsequent lectures, the slide engagement drops, reaching its lowest median w.r.t. the median slide engagement in the final lecture (NoSQL) where half the students using ASQ have the non-question slides visible or in focus less than 30% of the time. Again, we confirm prior works, e.g. [93], that have shown learners to be distracted by the availability of digital devices (laptops in particular). Even though ASQ offers a directed, continuous and structured laptop use during the course of a lecture, students are still distracted (which we infer from the fact that the ASQ window is hidden most of the time).

While the passive engagement allows us to determine how many students keep ASQ slides visible and/or in focus, we do not learn anything about those students (a significant percentage) that chose to not do so. One of our expectations is that students who engage passively (which means they are likely to not be distracted through Web surfing or similar activities) are also more likely to answer questions, as they followed the lecture. In order to evaluate this hypothesis, we compute for each student his or her *active engagement* which we define as the percentage of questions answered during a lecture.

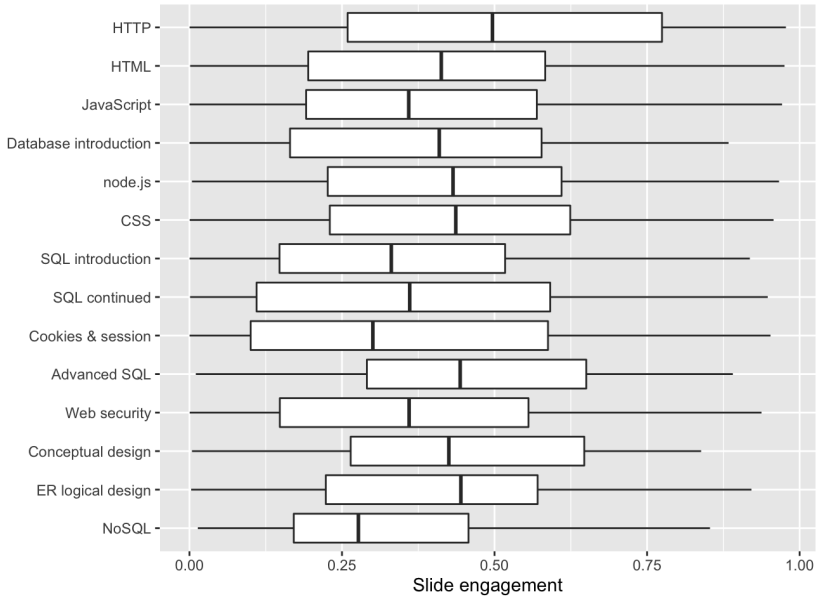


Figure 5.2. Distribution of slide engagement (computed per student) across all lectures. A value of 1 (0) indicates that ASQ was always (never) visible during class time.

For each lecture, we compute the Spearman rank correlation coefficient between students' question and slide engagement percentages. A correlation close to 1.0 would indicate that students with high slide engagement are also interacting with almost all questions. Table 5.1 shows the results of this analysis in the $\rho(S, Q)$ column. In the first lecture we observe a correlation of $\rho_{HTTP} = 0.72$ which indicates a strong relationship between slide and question engagement. In later lectures this relationship decreases in strength, reaching its lowest point in the final lecture with $\rho_{NoSQL} = 0.39$. This trend could be explained by the increasing complexity of the addressed topics, that might have discouraged students from participating in some or all of the question activities.

In contrast to previous studies, e.g. [77; 93; 188], we cannot compare our `ASQ`-using student population to the non-using population, as we only measure learning performance through questions posed within `ASQ`. Due to our privacy-aware setup, students attending lectures did not login to `ASQ`, nor did they provide their university IDs. As all question activities are formative assessments and students do not identify themselves when connecting to `ASQ` it is unclear why so many chose to not *attempt* to answer some of these questions. The lack of identification prohibits us to use the final exam score (as done in previous works) as learning performance measurement.

Although we *cannot* gather insights about students that chose to not consume lecture material through ASQ, we observed clear trends for students that did use the platform: slide and question engagement are strongly and positively correlated. We find this alignment of results an indication of the suitability of a platform like ASQ for teaching and experimental purposes.

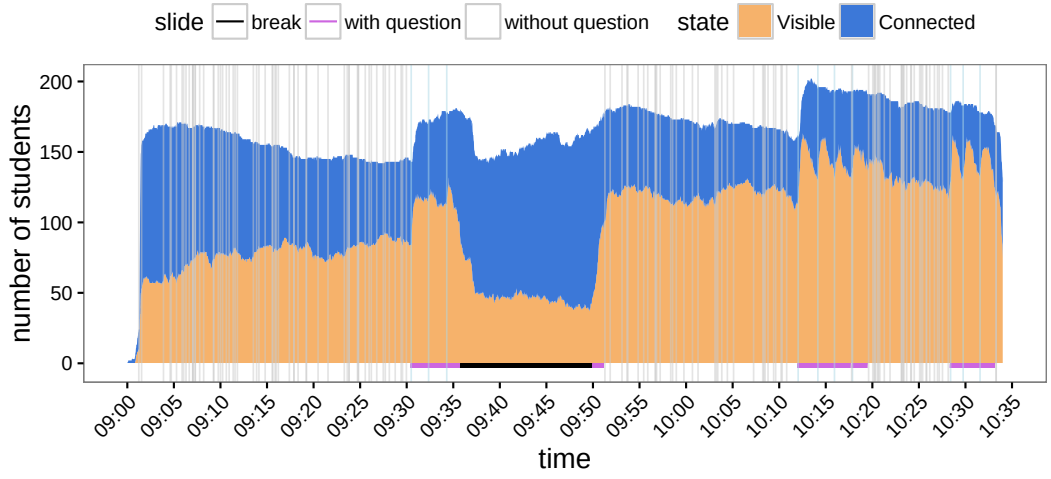


Figure 5.3. Temporal evolution of connected and slide engaged students during the HTTP lecture. Evidence of distracted behaviour in the second half of the lecture.

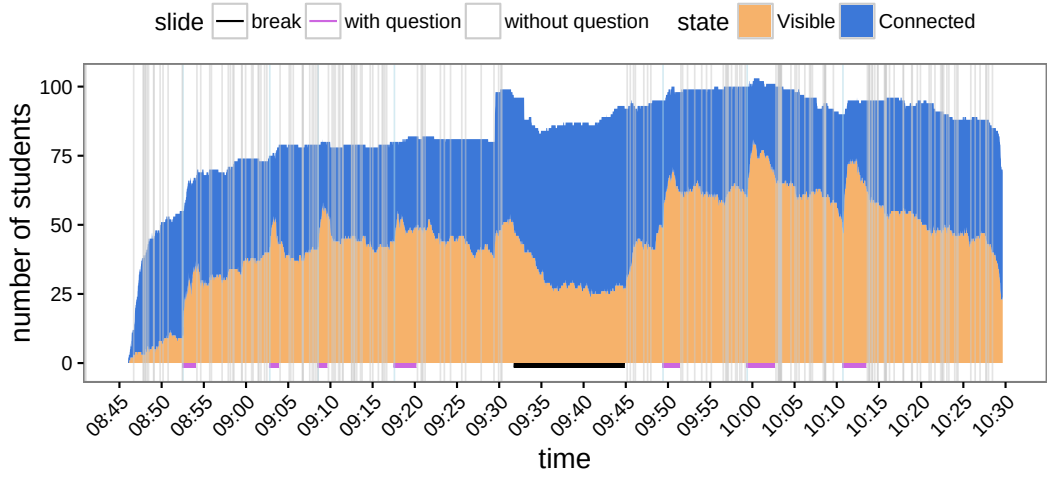


Figure 5.4. Temporal evolution of connected and slide engaged students during the CSS lecture. No clear slide engagement trend can be observed.

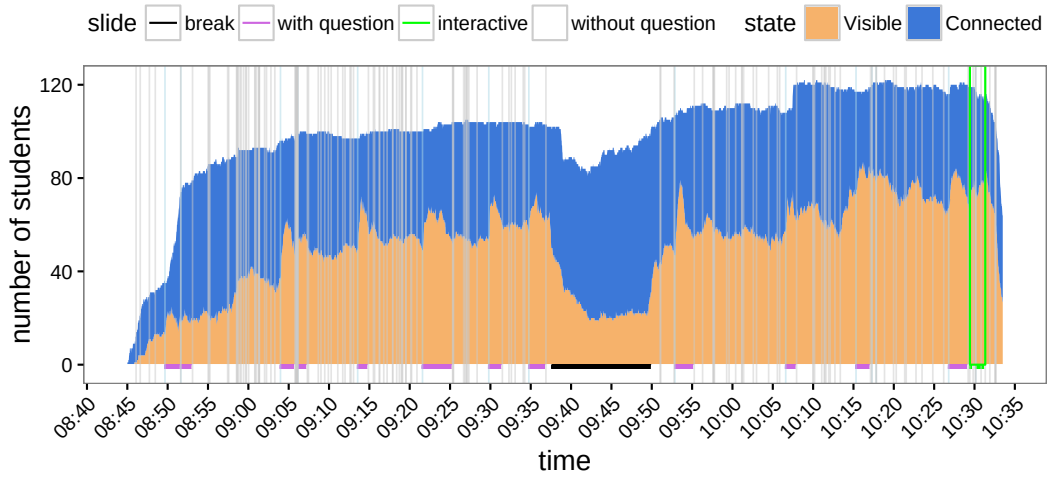


Figure 5.5. Temporal evolution of connected and slide engaged students during the Javascript lecture. Evidence of increasing slide engagement trend. Three interactive non-question slides are marked in green.

Lessons Learned

This section summarises what the authors of this case study that was published as [177] learned from the usage of ASQ in the classroom. The fourth and fifth authors are the instructors and designers of the *Web and Database Technology* course.

On the effectiveness of online questions to reduce distractions The results presented in the previous sections are mixed: while initially many students adopt ASQ, over the course of the class the adoption rate decreases. Although ASQ offers a directed use of a digital device to follow and engage with a large classroom lecture, a significant fraction of students are engaged elsewhere through browser activities (as also indicated in prior works). A qualitative inspection of the collected statistics show that the engagement of students with ASQ varies across lectures, and also within the same lecture. Take for instance the graphs in Figure 5.3–5.5; they show for each lecture second the number of students connected to ASQ (those are all students with ASQ either visible or hidden) and the number of students having ASQ visible on their screen. Javascript (Figure 5.5) is an example of lecture where it is possible to observe a growing engagement trend, both before and after the lecture's break, that is correlated with the progression of questions. During the CSS lecture (Figure 5.4), questions provided only local

increases in the number of engaged students, with no observable trend. Finally, lecture HTTP shows how the effect of questions changes during the lecture. Before the break, the burst of questions helped in increasing students' engagement. After the break, each peak due to a new question is immediately followed by a sudden drop, which indicates that students were prone to distract themselves after answering the question.

Interactive slides beyond questions In the JavaScript lecture (Figure 5.5), apart from the question slides a small number of additional slides contained interactive features to showcase JavaScript's abilities in the browser (such as a typing game and a text selection tool). However, when we explore whether those slides led to additional engagement (i.e. bringing students back from the other activities) we do not find this to be the case — for very few connected students the ASQ status changes from invisible to visible. This result is not supporting one of the authors' assumptions that more attractive/engaging ASQ slides lead to more direct student engagement.

5.2 Usability and Perceived Workload

A widely used subjective way to evaluate the usability of the system is a SUS questionnaire. The benefits of the scale include its short format (10 Likert scale items) that makes it quick to answer, and that it can be used to compare a variety of heterogeneous systems. The latter can also be deemed as a drawback since it makes the questionnaire very general. For each participant a score between 0 and 100 is calculated, which should not be interpreted as a percentage according to [156]. Sauro conducted of 500 different evalutions with more than 500 users and and the average score was 68. Anything above 68 can be considered to be above average while anything below 68 is considered below average.

5.2.1 Course Overview

We deployed ASQ in the 2015/16 edition of *Web and Database Technology*, a compulsory course for 1^{st} year BSc Computer Science and an elective for 3^{rd} year BSc minor students, at the Delft University of Technology. The course was followed by 310 students in total: 260 1^{st} year and 50 minor students. Across the eight course weeks, fifteen 90-minute lectures were given.

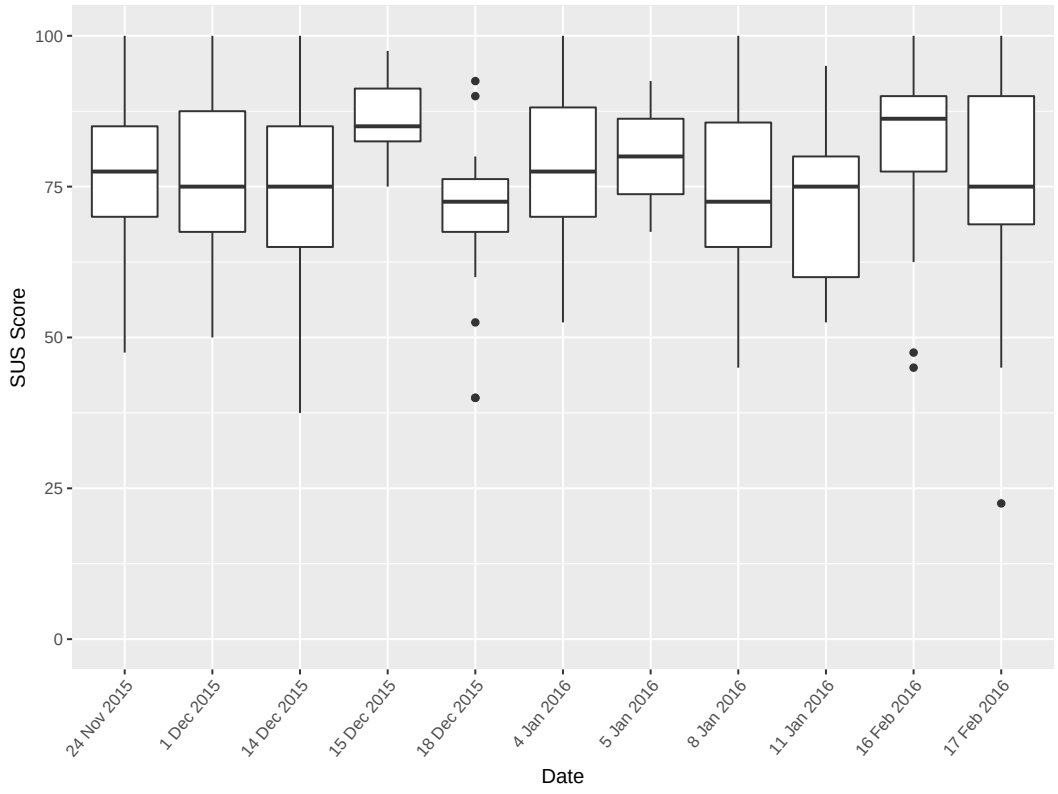


Figure 5.6. Boxplot of SUS scores for student responses

5.2.2 Usability

To evaluate the usability of ASQ from the perspective of both instructors and students we administered SUS [33] questionnaires to the participants (instructors and students of the course described in the previous subsection. The self-administered questionnaires were given to the participants immediately after the end of a lecturing session were ASQ was used. Three more questionnaires were administered in retrospective sessions on the 18th of December 2015 and 16,17 February of 2016. Participation in any of the questionnaire sessions was optional. Students consistently rated the platform above 68 with the average score across evaluations being 75.92. Although ASQ fared well in the evaluations, the number of students that participated in the study ranged from to 5% to 47% of all students that used ASQ during the lecture. Figure 5.6 displays a boxplot chart for the scores of each session. The mean scores for each session are summarized in Table 5.2.

Date	# of ASQ viewers	# of SUS responses	Mean SUS score
24 Nov 2015	89	42	78.21
1 Dec 2015	65	21	75.48
18 Dec 2015	–	19	69.47
14 Dec 2015	143	59	74.58
15 Dec 2015	19	6	86.25
4 Jan 2016	111	28	78.75
5 Jan 2016	29	2	80.00
11 Jan 2016	187	9	71.39
16 Feb 2016	–	36	82.15
17 Feb 2016	–	35	75.93
8 Jan 2016	83	16	75.78

Table 5.2. Mean SUS score of student responses per lecture session. "–" in the number of SUS responses denotes retrospective sessions where there was no lecture.

5.2.3 Perceived Workload

While ASQ strives to scale active learning to large audiences, nevertheless the tool will impose extra workload to instructors that use it. To gauge whether this load is within acceptable levels, we asked the two instructors of the course described in subsection 5.2.1 to complete a NASA-TLX[90], a subjective, multidimensional assessment tool that rates perceived workload. In the first part of the NASA Task Load Index (NASA-TLX) questionnaire subjects are asked to rate their experience in six subjective subscales: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort and Frustration. In the second part, subjects compare pairwise the subscales of the first part in order to create an individual weighting. The number each subscale is chosen over another is the weighted score of this subscale and it is multiplied by the scale score for each dimension and then divided by 15. The result, a workload score that ranges from 0 to 100, is the overall task load index.

Table 5.3 shows the overall task load index scores for 13 of the course lectures where the questionnaire was administered. Both the instructors had higher workload when they first used ASQ (may be due to in-familiarity with the interface) which decreased with the progression of the course. Specifically in the first lecture (HTTP) the workload is very high because the cockpit view which shows the preview of the next slides did not work. Even the average workload for both the instructors through out the course was lower (around 30 and 40) indicat-

ing that ASQ can be used by teachers with ease after going through a learning period.

Lecture	Instructor	Overall Task Load Index
HTTP†	$I1$	68.66
HTML	$I1$	28.66
JavaScript	$I1$	36.33
Database introduction	$I2$	62.33
CSS	$I1$	36.00
SQL Introduction	$I2$	32.66
SQL Continued	$I2$	28.33
Cookies & Session	$I1$	35.33
Advanced SQL	$I2$	32.33
Web Security	$I1$	56.33
Conceptual Sesign	$I2$	23.66
ER Logical Sesign	$I2$	20.66
NoSQL	$I2$	14.00

Table 5.3. Mean SUS score of student responses per lecture session
† Cockpit view did not work for next slide preview

5.3 Summary

Our main objective in this chapter was to establish the suitability of ASQ as platform for "in-situ" privacy-preserving research in education. To do so, we first discussed results on student engagement inferred from the event logging capabilities of ASQ from a longitudinal study over 10 weeks of lectures, operating under strict *privacy* and *fairness* requirements, both obvious ethical concerns in real-world courses.

In a similar setting to the one described above, we looked into the usability and workload aspects of the application. We reported positive results on SUS and NASA-TLX questionnaires that were given to students and instructors respectively. This further illustrates the potential ASQ to be integrated as a teaching tool in real-world classrooms. However, ASQ participation in lectures was descreasing as the course progressed and the majority of students that participated in a lecture did not provide us with their feedback. This makes us cautious about our claims and calls for more exhaustive studies in the usability aspect of the tool.

In the next chapter, we will look into a case study that takes advantage of the low-level events of ASQ to create higher level attention states that can provide instructors with insights about the classroom attention.

Chapter 6

Case Study: Inferring Student Attention

6.1 Introduction

In subsection 2.3.1 we presented a broad body of research findings that highlight the increased episodes of inattention occurring during post-secondary classroom-based learning and suggested that interactive elements can have a positive impact in capturing student attention. We also commented on the difficulty of employing most of the techniques to gauge attention in large classes due to prohibitive costs, their obtrusive nature and/or their inability to determine students attention in real-time.

In the case study of this chapter, we investigate to what extent modern Web technologies can facilitate and enable the *continuous*, *scalable* and *unobtrusive* inference of student attention *in real-time*. We target the traditional classroom setting – so as to enable lecturers to *react* in a timely manner to the attention needs of their students – and we focus on Web-mediated teaching and formative assessment activities. We seek to answer to the following Research Questions :

RQ1.2 *To what extent can students' attention be inferred from their interactions with a Web-based interactive presentations application that facilitates in-lecture active learning questions practice?*

RQ1.3 *Which type of interactions are most correlated with (in)attention?*

As common in previous works, we infer attention from students' retention levels. To this end, we will utilize the extensive logging capabilities of `ASQ` that enable the tracking and recording of real-time students' interactions during lectures. We deployed `ASQ` in the context of three ninety minute university-level lectures given by two different instructors, with varying interactivity levels and up to 187 students. Our results show that `ASQ` can provide fine-grained insights on students' attention states that relate to previous findings on the subject, thus

Table 6.1. Overview of activity indicators based on browser events.

Name	Description
`exercise`	True when the current slide has an exercise.
`connected`	True when the student browser is connected.
`focus`	True when the browser has focus on the tab or exercise related to the lecture.
`idle`	True from the time of an idle event until one of `tabhidden`, `tabvisible`, `windowfocus`, `windowblur`, `focusin`, `focusout`, `exercisefocus`, `exerciseblur`, `input`, `questioninput`, `exercisesubmit` and `answersubmit` occurs.
`input`	True when an `input` or `questioninput` event occurs. This state is valid only on slides that contain exercises.
`submitted`	True when the student has submitted at least once this exercise (as indicated by an `exercisesubmit` event). This state is valid only on slides that contain exercises.

demonstrating `ASQ`'s ability to obtain an accurate view of students' attention in a classroom setting.

6.2 From Low-level Events to Attention States

Recall that our overarching goal is to infer student attention. To this end, based on the introduced low-level events of section B.1.3, we define higher-level activity indicators, which denote the activity (or lack thereof) currently performed by a student in a lecture. Subsequently, we use these indicators to infer a basic model of *student attention states*.

Activity indicators

Each low-level browser event occurs at a specific point in time; we map sequences of browser events generated by a student to one of six binary activity indicators, which we consider to be natural components of a student's attention state. These indicators are non-exclusive (i.e. several indicators can be true at the same time) and listed in Table 6.1: `exercise`, `connected`, `focus`, `idle`, `input` and `submitted`.

Student attention states

We take a data-driven approach to the exploration of the activity indicators and in Table 6.2 list all the 17 combinations of indicators that we observed in our

Table 6.2. Modeling student attention based on activity indicators. Activity indicators are binary, ✓ represents `True`, and - represents `False`.

exercise	connected	focus	idle	input	submitted	Inferred Attention State
-	-	-	-	-	-	Disconnected
✓	-	-	-	-	-	Disconnected
✓	-	-	-	-	✓	Disconnected
-	✓	-	-	-	-	Distracted
-	✓	-	✓	-	-	Distracted
✓	✓	-	-	-	-	Searching for a solution
✓	✓	-	✓	-	-	Searching for a solution
-	✓	✓	-	-	-	Interacting with non-question slide
-	✓	✓	✓	-	-	Following
✓	✓	✓	-	-	-	Thinking
✓	✓	✓	✓	-	-	Thinking
✓	✓	-	-	-	✓	Bored
✓	✓	-	✓	-	✓	Bored
✓	✓	✓	-	-	✓	Waiting
✓	✓	✓	✓	-	✓	Waiting
✓	✓	✓	-	✓	-	Working on an answer
✓	✓	✓	-	✓	✓	Reworking answer

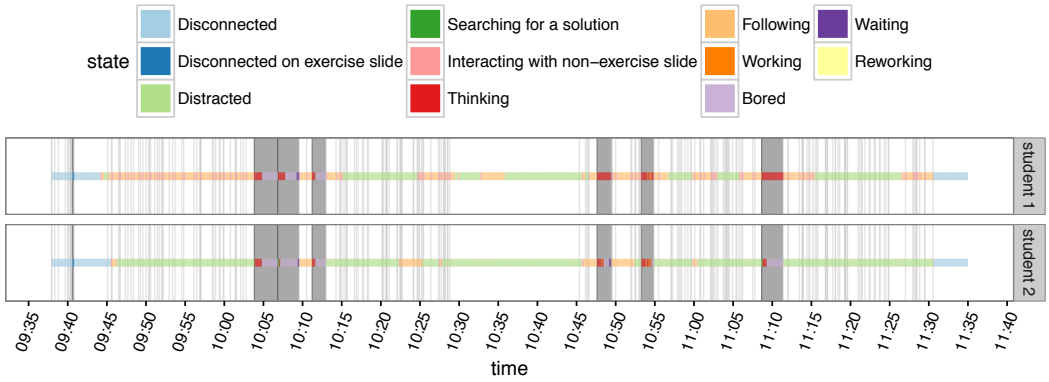


Figure 6.1. Two example progressions of inferred attention states during the course of a single 90-minute lecture (specifically: Web Security). The dark-grey areas represent slides with interactive exercises (6 in total), while the light-grey vertical bars indicate slide transitions. While Student 1 starts off highly attentive, Student 2 is inattentive from the start of the lecture.

data traces (described in detail in Section 6.3). We manually assign ten different semantic labels to each combination. For instance, a student who has `submitted` an answer to an `exercise` and is now `idle` with `ASQ` in `focus` is considered to be *Waiting* (e.g. for the instructor to provide feedback), while a student who also `submitted` an answer and is neither `idle` nor having `ASQ` in `focus` is considered to be *Bored* (and occupying himself with other activities on the device). Thus, at each point in time a student is in exactly one of the ten listed attention states. Figure 6.1 showcases the progression of two students' attention states across an entire lecture; while *Student 1* starts off the lecture at a high level of attention (indicating by the continuous *Following* state) and later on toggles between the *Following* and *Distracted* states, *Student 2* starts off the lecture in a *Distracted* state and only exhibits short bursts of attention shortly before or after some of the interactive exercises.

Although we are using psychological terms such as *Bored*, *Distracted*, *Thinking*, and the like, these should be not be interpreted beyond the strict definition of Table 6.2 as our goal is to give a readable representation of the aggregated activity indicators that can be amenable of further analysis and experimentation. In the remainder of this case study we analyze to what extent our definition of inferred attention states is suitable to reproduce findings from the literature.

Figure 6.2. ASQ in the classroom: most students' laptops are connected and focused on the slide material being explained.

6.3 **ASQ** Deployment & Data Collection

This experiment took part during the same deployment described in subsection 5.2.1. We report on results in three of the sessions were ASQ was used, identified as suitable for experimentation: at regular intervals, the lecture material was interspersed with interactive elements consisting of live programming exercises, multiple choice questions, and visual question types. Figure 6.2 shows an example session of ASQ in the classroom.

At the beginning of each ASQ session, students in the lecture hall were instructed (though not compelled) to open the lecture presentation in the browser. Students connected anonymously; a random identifier was assigned to each connection, enabling us to group all interactions made by the same student within one lecture together (identifying markers *across* lectures were not retained for privacy reasons). The lecture slides were not only visible in the students' browser but also on the lecture hall screen and thus students who decided not to use ASQ treated the sessions as standard lectures.

We posed questions of three question types that depended on the lecture material and assessment goals of each class: (A) multiple-choice, (B) SQLite programming (Figure 6.3), and (C) text-input (Figure A.10). Table 6.3 summarizes the main characteristics of the three lectures, including the lectures' topic,

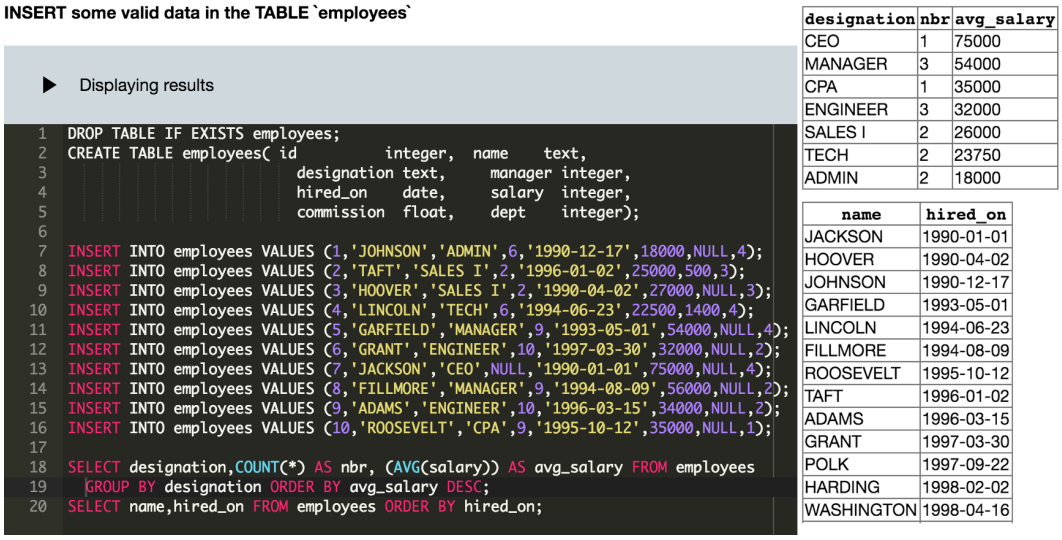


Figure 6.3. SQLite question from Lecture 1, Advanced SQL. It comprises a text editor (left) to write and execute SQL queries on an in-browser database instance, and a results pane (right) to visualize the query results.

the number of students participating through `ASQ` and the number of questions posed per type. Note that *Lecture 1. Advanced SQL Topics* has generated almost seven times more browser events than the other two lectures due to its usage of SQLite programming quizzes: not only the large amount of typing contributed to the events generation, but also the question setup which required the students to consult a database schema diagram resulting in a considerable amount of blur/focus events between `ASQ` and the diagram.

6.4 Analysis

In our exploration of the collected logs, we are guided by our research questions and the five main findings of prior works (identified in subsection 2.3.1 exploring students' attentiveness in the classroom).

F1: Students' attention drops over the class period. For all lecture logs, we translated low-level browser events into activity indicators (Fig. 6.4) and subsequently inferred attention states (Fig. 6.5). We consider the two activity indicators `connected` and `focused` and the union of the states *Following/Thinking/-Working* as well as *Distracted/Bored* as most suitable representatives of student attention and inattention respectively. To explore how attention changes over

Table 6.3. Overview of the three ASQ lecture sessions each given by one of two instructors (identified as I1 and I2). For each session, the number of students participating, the number of exercises (per type) and the number of ASQ low-level browser events logged are listed.

	Instr.	Topic	#Students using ASQ	#ASQ events	#Question types A	B	C
1	I1	*Advanced SQL Topics*	143	121,062	0	7	0
2	I1	*ER Conceptual Design*	111	17,460	8	0	0
3	I2	*Web Security*	187	17,562	4	0	2

time, we correlate the lecture time (in units of 1 second) with the number of students in the specific state(s) or activity setting. If, as expected student attention drops over time, we will observe a decrease in `focus` over time and an increase in *Distracted/Bored* students. The results in Table 6.4 show that this is indeed the case: inattention-oriented activities/states are positively correlated with time while attention-oriented activities/states are negatively correlated with time. Moreover, the high-level inferred attention states achieve higher absolute correlations, indicating that they are more suitable to infer (in)attention than our low-level activity indicators.

We thus posit that based on the events logged in ASQ, we are able to infer in real-time (and live in the classroom) when and to what extent attention drops over time, relying on either the `focus` activity indicator as a basic measure or a combination of the more high-level attention states *Following/Thinking/Working* (and their counterparts).

F2: Attention breaks occur regularly and increase in frequency as the class progresses. For each second of the lecture we track the number of *attention breaks*, that is, the number of students that switch their device from `focused` on ASQ to some other activity. We also track *attention recovery* which we define as the number of students whose device switches back to `focus` on ASQ. The *attention focus variation* is the net sum of attention recoveries minus the attention breaks observed during the same period (a window of 30 seconds). For each of the three lectures we present their *attention focus variations* in Figure 6.6. We observe that attention breaks occur regularly but there is no noticeable increase in frequency as the class progresses. We note that although this is in contrast to $F2$, not all empirical studies in the past observed this increase in attention breaks [173].

Table 6.4. Linear correlation coefficient (significant correlations at the $p < 0.05$ level are marked †) between time and number of students exhibiting a particular activity indicator (top part) or one of a set of inferred attention states (bottom part).

		+++ Activity indicators +++			
		All slides		Slides w/o exercises	
	Lecture	Connected	Focus	Connected	Focus
1	Advanced SQL Topics	0.176†	-0.182†	0.281†	-0.059
2	ER Conceptual Design	-0.224†	-0.569†	-0.284†	-0.637†
3	Web Security	0.263†	-0.177†	0.284†	-0.228†
		+++ Attention states +++			
		All slides		Slides w/o exercises	
	Lecture	Distracted/ Bored	Following/ Thinking/ Working	Distracted	Following/ Thinking
1	Advanced SQL Topics	0.324†	-0.274†	0.450†	-0.257†
2	ER Conceptual Design	0.039	-0.549†	0.230†	-0.657†
3	Web Security	0.391†	-0.262†	0.458†	-0.390†

F3: Attention rises when interactive elements are introduced. Drawing on our analysis of attention focus variation, we observe that whenever there are interactive elements in the slide, in the form of questions, we observe spikes of attention recovery (Fig. 6.6) and an increase of connected students (Fig. 6.4). While introducing interactive elements thus captures the attention of the students (positive attention focus variation), shortly thereafter we observe the subsequent loss of focus due to students waiting on each other to answer. Likewise, students might be searching for solutions using their devices, something `ASQ` cannot distinguish from students simply leaving the application to do something else. As we can observe in the charts of Fig. 6.5 for all the lectures, whenever there is a slide with a question, the number of students that have their `ASQ` page out of focus (`Searching` state) is always lower than in slides without a question (`Distracted` state). Similarly, the magnitude of attention focus variation is smaller for slides without questions than for slides with questions, which literally appear to send jolts through the collective attention span of the students in the classroom (Fig. 6.6). Our results thus confirm previous findings of rising attention at interactive elements.

F4: Immediately after interactive teaching elements, the level of distraction is lower than before the start of the interaction. While there is a peak of interest as soon as questions are asked, after students submit their answers, their focus switches to other activities. Hence, as shown in Fig. 6.5 towards the end of the question, the number of students we infer to be in a *Distracted* state rises considerably and is almost always higher than right before the interactive teaching element. The effect depends on the length of time students have to wait for other students to complete the exercise (before the instructor moves on in the lecture) and on the type of feedback given either individually or globally on the submitted answer. This result is a clear deviation from prior works and suggests that our attention model, in particular the *Distracted* state captures more than just students' distraction.

F5: In retention tests, students tend to perform better on material presented early on in the class. Instead of dedicated retention tests, we rely on the multiple choice (MC) questions as a retention proxy (we restrict ourselves to MC questions as the open question types require manual grading to achieve highly accurate results).

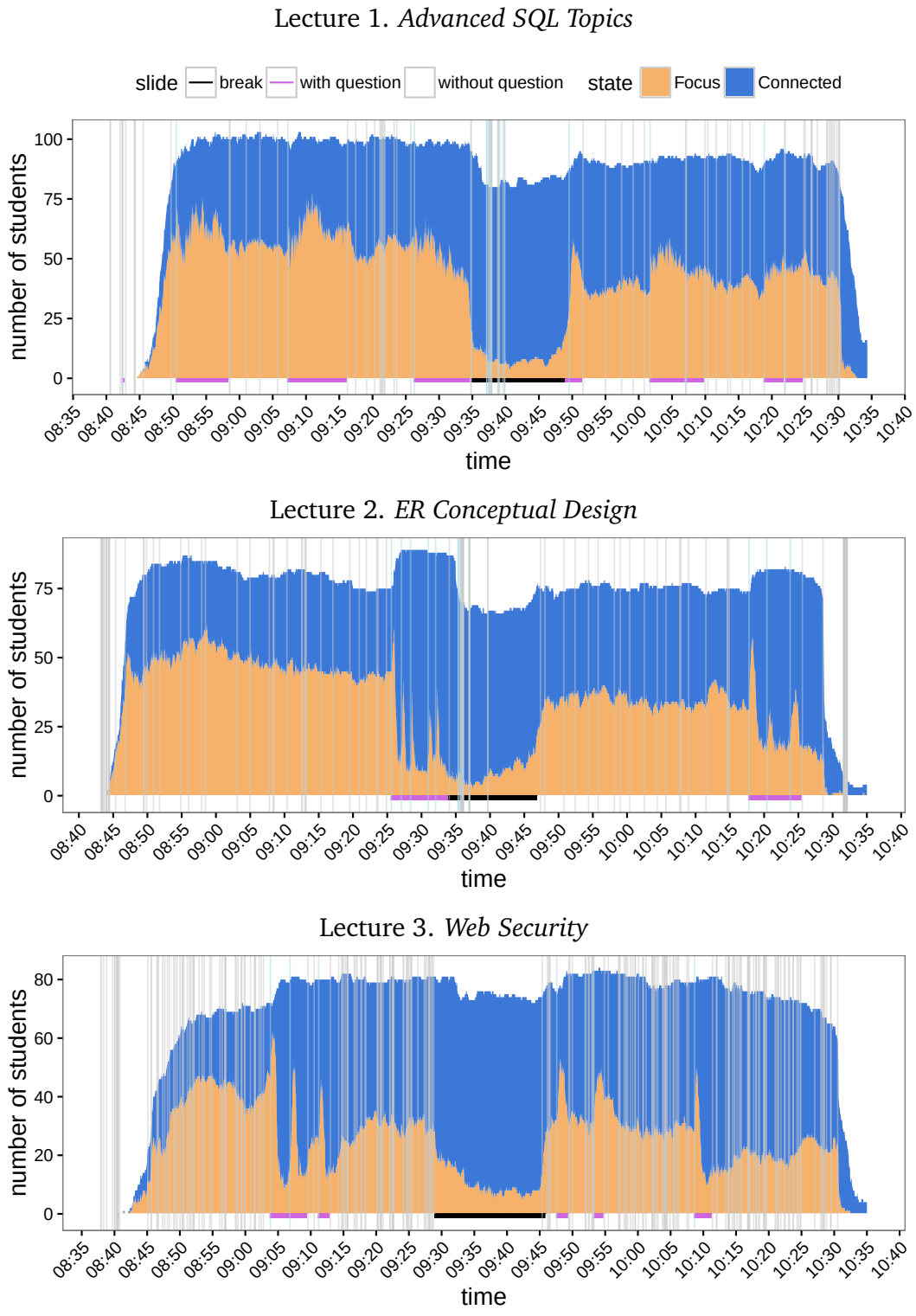
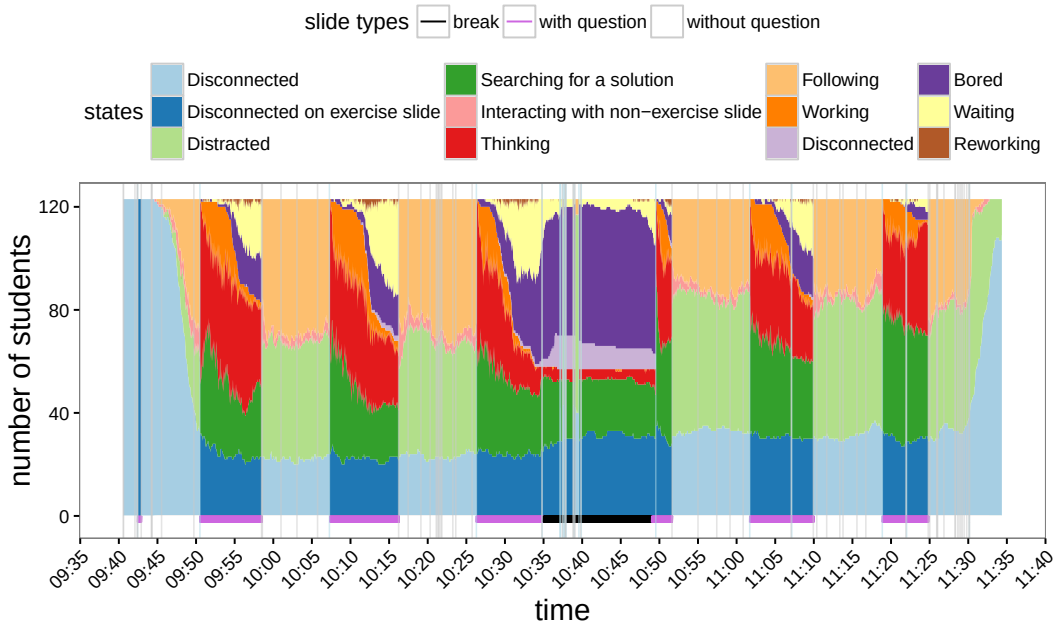
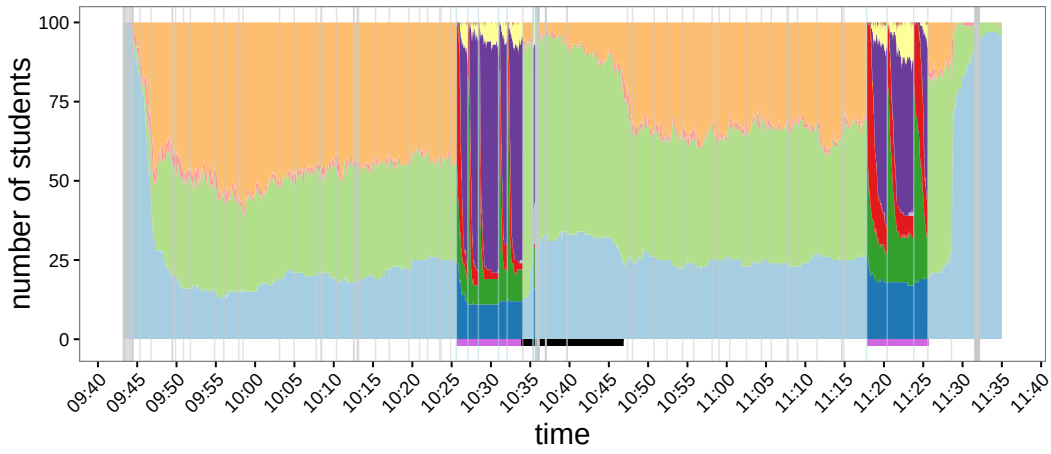


Figure 6.4. Connected and Focused activity indicators for all the sessions

Lecture 1. *Advanced SQL Topics*



Lecture 2. *ER Conceptual Design*



Lecture 3. *Web Security*

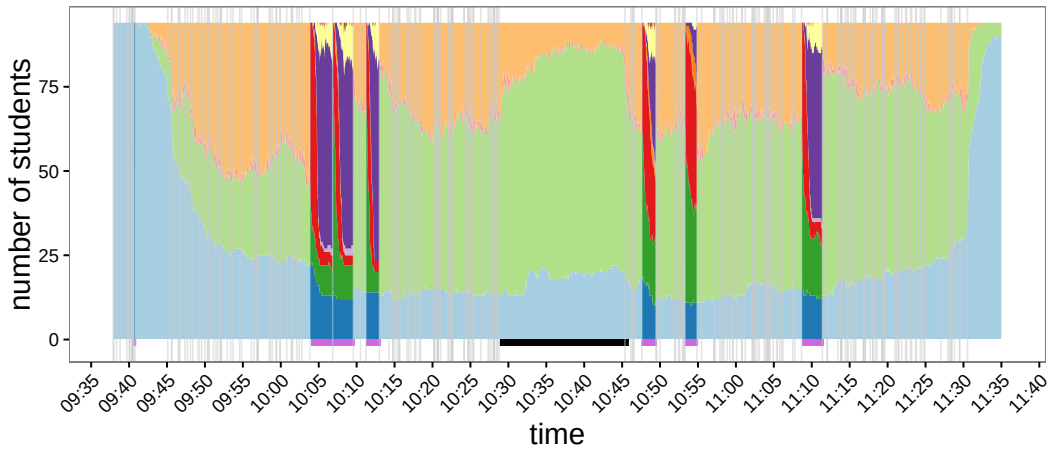
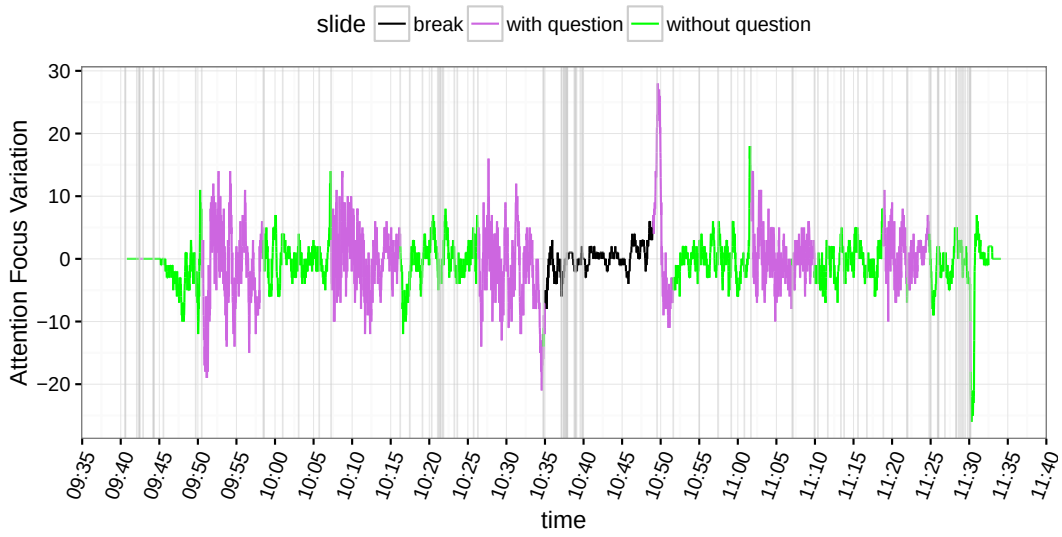
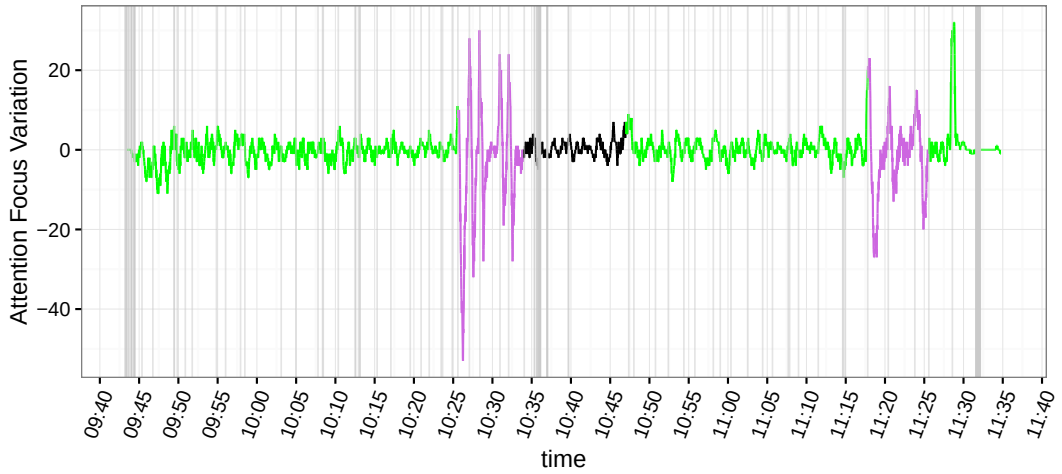


Figure 6.5. Inferred student attention state for all the sessions

Lecture 1. *Advanced SQL Topics*



Lecture 2. *ER Conceptual Design*



Lecture 3. *Web Security*

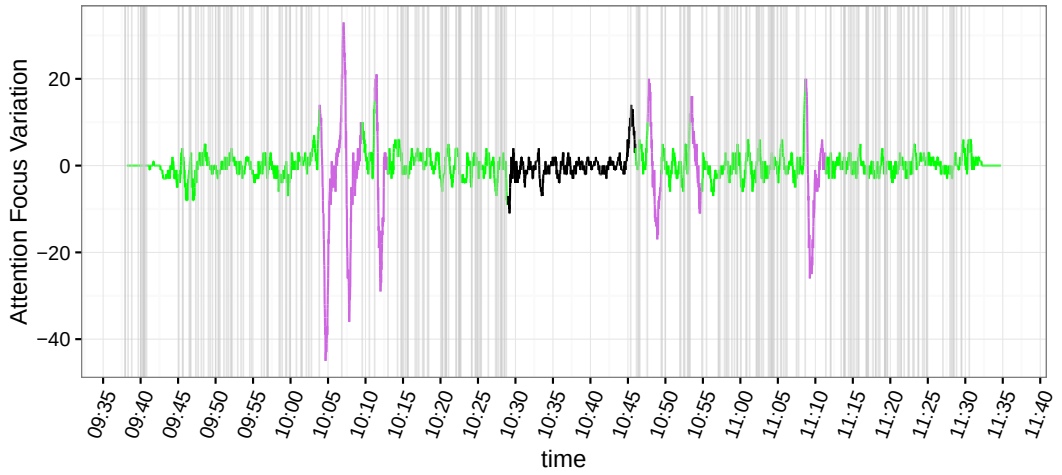


Figure 6.6. Attention Focus Variation: how many students have changed the focus of attention during lecture (moving sum of attention breaks and recoveries using a window of 30 seconds).

Table 6.5. Correct vs incorrect answers ordered by time of question for Lecture 2

question	start time	correct%	correct	incorrect	total
1	10:25:36	92.31	72	6	78
2	10:27:04	77.22	61	18	79
3	10:28:24	82.50	66	14	80
4	10:30:58	1.43	1	69	70
5	10:32:04	90.79	69	7	76
6	11:17:48	70.77	46	19	65
7	11:20:25	75.41	46	15	61
8	11:23:48	65.00	39	21	60

Table 6.5 lists the accuracy of the student answers for the eight MC questions of *Lecture 2. ER Conceptual Design* as well as the specific time they were posed in the lecture. Note that shortly after 10:30am the official 15 minute break commenced. We observe that students tend to perform better in the first half of the class than the second. Although a subset of questions from a single lecture do not provide enough evidence to support or reject this finding in the context of ASQ it shows once more ASQ's capabilities to provide fine-grained real-time logging and analyses to the instructor.

6.5 Summary

ASQ is able to provide *real-time data analytics in the classroom*, thus providing lecturers with the capability to observe their students level of engagement in a data-driven manner. We do so as students follow the material being presented (slides) but also as they interact with the exercises and various types of questions embedded in some of the slides.

In this case study we have shown how ASQ can be employed to infer student attention, based on either activity indicators and states we aggregate based on low-level browser events. The visualizations presented here enable instructors to observe at a very fine-grained level the behavior of an entire class with hundreds (or potentially thousands) of students. Our analysis confirms existing research findings, whereby: 1) student attention drops during the class period; 2) attention breaks occur regularly as the class progresses; and 3) attention rises when interactive elements are introduced. Additionally, we could observe a drop in attention as soon as the interactive activity is completed by individual students, which should be taken into account when planning to introduce questions and

interactive exercises within a lecture.

For in-class realtime feedback on students' attention, the graphs with the high level attention states maybe overwhelming for instructors. It is worth pursuing compacting the information that that ASQ provides instructors based on the current teaching context. For example, during questions the tool may display a progress bar like the one presented in subsection A.3.1. The progress bar displays the number of students whose state is "true" for one of the following 4 activity indicators: *submitted*, *working*, *focus* and *idle* (in order of importance). If one student has more than one of the activity indicators set to "true", the tool just displays the most important one. Consider the case where a student has both submitted an answer and is idle. The answer submission is the most important information for the instructor in this case, so the tool will just updated the number of students in the "submitted" activity indicator.

The monitoring of student behavior can, of course, be utilized to optimize questions. In the next chapter, we will see a relevenat example in a case study where we observe student engagement patterns to draw conclusions on strategies for temporal practice question placement in lectures.

Chapter 7

Exploratory Study: Comparing Eye-tracking Gaze Data With the Visibility **ASQ** Activity Indicator

The attention states we presented in chapter 6 are computed as a combination of the various activity indicators (presented in section 6.2) that we compute based on the events collected by `ASQ`. Due to the nature of these activity indicators we just have an approximation of the behavior of each student that correlates well with findings in student attention. A promising way to explore exactly how accurate is this approximation is to compare the `ASQ` activity indicators with eye tracker data on visual attention [63, p. 11] from students that use `ASQ` in lectures. Specifically, we will explore how the `visible` activity indicator the describes the visibility of the `ASQ` browser window relate to fixations of the gaze of the students on the screen under various conditions.

We aim to answer the following question:

RQ1.4 *How does student gaze behavior as recorded through the use of eye-tracking relates to activity indicators generated with the aforementioned Web application?*

To answer this question we conducted an experiment that took place during the lectures of a course in functional and logic programming. The subjects' workstations were equipped with eye-tracking devices that provided us with gaze data and more specifically fixations of the students' gaze. We present our analysis results which show that, in general, students spent a small portion of their visual attention towards looking at the `ASQ` window when it was visible. However, increased visual attention patterns emerged during slides that contained questions compared to slides that did not and during times when the students

was performing input on a question compared to times when they had submitted a response.

7.1 Experimental Setup and Methodology

Table 7.1. Overview of the lectures in the case study. Lectures are reported in temporal order of delivery. The lectures were divided into two separate sessions (e.g., Prolog 1a and Prolog 1b), i.e., the same content was lectured to two different groups of students. Legend. #ASQ: number of students connected to `ASQ`. #Q: number of questions in the lecture slides. #A: number of total answers in the lecture. %SQ: the percentage of students connected to `ASQ` and answering at least one question (≥1 question submitted). %75Q: percentage of students connected to `ASQ` and submitting an answer to at least 75% of the questions in the lecture. %AQ: percentage of students connected to `ASQ` and submitting an answer to all questions. %SN: percentage of students' initiated inputs to the questions that resulted in no submit.

Lecture	**#ASQ**	**#Q**	**#A**	**%SQ**	**%75Q**	**%AQ**	**%SN**
Prolog 1a	27	20	446	88.89	74.07	3.70	10.99
Prolog 1b	22	23	475	95.45	77.27	9.09	10.55
Prolog 2a	23	26	469	82.61	52.17	8.70	25.20
Prolog 2b	22	26	483	81.82	59.09	22.73	20.43
Prolog 3a	19	11	209	94.74	73.68	31.58	15.73
Prolog 3b	20	10	220	100.00	80.00	50.00	20.86

7.1.1 Course

The experiment took place during the second half of the lecture of the "Functional and Logic Programming" (FLP) course at the Slovak University of Technology in Bratislava in the spring semester of 2016. The course is elective and taught at the bachelor level of study. It is organized bi-annually and is usually taken by students of the 2nd and 3rd year. The first half introduces functional programming through the LISP and Python languages. Logic programming is introduced through the Prolog language during the seconds half. Successful completion of the course rewards students with 6 ECTS credits. The course lasts for 12 weeks. Its format comprises weekly lectures (100 minutes each), laboratory practices

(100 minutes each) as well as individual project activities. It was attended by 46 students. Table 7.1 gives an overview of all the lectures.

After performing an exploratory analysis of the obtained data, we found out there were some technical problems during the recording of the last session. Moreover, there were missing data from the first session. This force us to exclude the lectures "Prolog1a" and "Prolog03b" from our analysis. Table 7.2 shows how many complete records from subjects we obtained in every session: these are the data on which we will perform our analysis.

Table 7.2. Number of complete session records for each session.

Session	**# of complete records**
Prolog01b	17
Prolog02a	18
Prolog02b	16
Prolog03a	19

7.1.2 Experimental Setup

As part of the second half of the course, a single instructor had to give three lectures explaining concepts of the Prolog programming language . Each lecture was less than two hours long. The lectures were given in two groups at the room of the User Experience and Interactions Research Center of the faculty due to the limited capacity of 20 working stations. All the working stations are equipped with an *Tobii X2-60 eye-tracker*, with 60 Hz sampling frequency to track the gaze of each student subject. Moreover, screen activity was captured on a video and the subjects's face was capturing using a Creative Senz3D depth camera. We also captured three different sets of events: 1. Operating System (OS) events 2. browser events using a custom browser extension for the Chrome browser 3. Input/Ouput (I/O) events .

Finally, we recorded a video of each lecture with a tripod camera situated in back (slightly on the right) of the room.

`ASQ` was used to present examples ASQ and pose questions. The instructor decided to use six question types, five of which could be automatically assessed for correctness. The list of question types is as follows:

1. **multi-choice**. Multi-choice question (auto-assessed)
2. **short answer**. Single-line text input (auto-assessed)

3. **highlight the text** (example in Figure 7.1). Highlight portions of text using the appropriate color (auto-assessed)

4. **classify**. Classify the label dropping it into the correct bucket (auto-assessed)

5. **order**. Place items in the correct order (auto-assessed)

6. **code**. Code editor with syntax highlighting supporting many programming languages. In this specific course, we used tasks in LISP and Prolog.

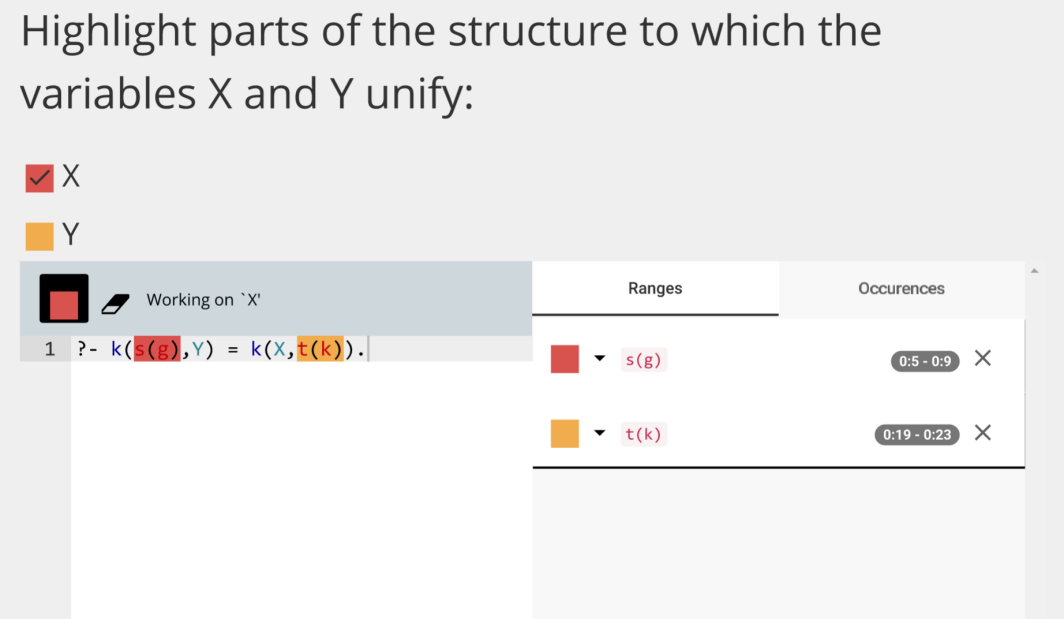


Figure 7.1. Example of a highlight question posed in the lectures Prolog 2a and Prolog 2b.

7.1.3 Methodology

In this study, we will report on the correlation between the `ASQ` inferred activity indicators `visible` and `focus` the were presented in section 6.2 and the equivalent indicators that we will map to eye-tracker data. We will use fixations – times when the gaze is held in a single location – as they are considered to be a good measure of visual interest [63; 34])

Eye-tracker Data and Preprocessing

The key objective was to distinguish between records where the subject gaze was fixated out of the screen and records where the gaze was fixated inside it. The first step towards our objective was to use the validity codes provides by the Tobii device to keep valid records and discard the invalid ones. The format of the records was an integer from from 0 to 4 for each eye. Records with a value of less than 2 for both eyes were rejected as per the Tobii manual of the eye-tracker device ([163]). The next step was to compute gaze coordinates for each record. If only one eye was valid we used date from this eye only to determine the gaze. The final step was to to classify fixations based on whether they were within the screen bounds or out of them by assigning them the "fixation_onscreen" and "fixation_outofscreen" classes. We marked a record as in screen when the fixation position was in the interval <0;1>. The rest valid fixations records were assigned "fixation_outofscreen" class.

7.2 Results

Lecture	fixation on-screen		fixation off-screen	
	visible	hidden	visible	hidden
Prolog01b	31%	4%	63%	2%
Prolog02a	38%	8%	50%	3%
Prolog02b	26%	6%	62%	6%
Prolog03a	16%	17%	49%	19%
All	**28%**	**8%**	**57%**	**7%**

Table 7.3. Summary – for all lectures and all slides – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen the `ASQ` window *visible* and `ASQ` window *hidden* states captured by `ASQ`.

We begin our exploration by looking at how the visibility of the `ASQ` window – *visible* or *hidden* – is related to the fixation of eye-tracking-overview-activethe students *on* or *off* screen that displays the `ASQ` window across all lectures.

Table 7.3 shows the mean percentage of all the combinations for all students in each lecture and the aggregated mean in the bottom. On average, when the `ASQ` window is visible students spent 28% of the time in a lecture looking at it and another 57% not looking at it.

Lecture	fixation on-screen		fixation off-screen	
	visible	hidden	visible	hidden
Prolog01b	16%	22%	52%	10%
Prolog02a	22%	22%	49%	7%
Prolog02b	11%	16%	60%	13%
Prolog03a	4%	39%	12%	45%
All	**15%**	**23%**	**48%**	**14%**

Table 7.4. Summary – for slides without a question in all lectures – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen the `ASQ` window *visible* and `ASQ` window *hidden* states captured by `ASQ`.

Lecture	fixation on-screen		fixation off-screen	
	`ASQ` visible	`ASQ` hidden	`ASQ` visible	`ASQ` hidden
Prolog01b	32%	3%	64%	1%
Prolog02a	41%	6%	50%	2%
Prolog02b	28%	5%	62%	5%
Prolog03a	16%	15%	51%	17%
All	**29%**	**7%**	**58%**	**6%**

Table 7.5. Summary – for slides with questions in all lectures – of the percentage of time that the students' gaze during a lecture is fixated on or off the screen during the `ASQ` window *visible* and `ASQ` window *hidden* states captured by `ASQ`.

To get a better understanding in how this translates to different types of slides we broke down the data in two categories: those that occurred during slides that did not contained questions and those that did. In the first case (see Table 7.5). We see that the fixation on-screen when the `ASQ` window is visible decreased by 14% to 15% and the gaze on screen, again when the `ASQ` window as visible, by 9% to 48%. During questions (Table 7.5) when the `ASQ` window is visible, both the on-screen and off-screen fixations increase by 1% to 29% and 58% respectively. Figure 7.6 shows a more intuitive distribution of how students spend their time in the aforementioned cases.

When other activity indicators are taking into consideration for the two types of slides we notice some interesting behavior as shown in Table 7.6. During question slides when students are working, as indicated by the *input* activity indicator, the students are looking at the `ASQ` window 68% of the time. During the rest 32% the gaze fixations are out of the screen but the `ASQ` window is still

visible. Probably this translates to activities related to the exercise like looking at notes, discussing with the teacher or their classmates or looking at a whiteboard. However once they submit their answer (*submit* activity indicator) they look at the ASQ window 20% of the time, spending most of the rest at gazing off-screen with the ASQ window visible. The 5% difference of fixation on-screen/visible between this condition and slides without question in general maybe due to the fact that students are still thinking about their submission. Finally, when the students are idle (no browser activity for more than 10 seconds) during non question slides the on-screen/visible combination reaches, as expected, its lowest value of 14%. A considerable amount of time (22%) is spend looking at off-task material in the screen.

Slides	**Condition**	**fixation on-screen**		**fixation off-screen**	
		visible	**hidden**	**visible**	**hidden**
WQ	input	68%	0%	32%	0%
WQ	submitted	20%	11%	59%	10%
NQ	idle	14%	22%	49%	15%

Table 7.6. Summary – for different conditions across all lectures – of the percentage of time that the students' gaze during a lecture is fixated is on or off the screen the ASQ window *visible* and ASQ window *hidden* states captured by ASQ. **Legend**. *WQ*: Slides that contained questions. *NQ*: Slides without a question. We report the following conditions: *input*: learners were giving an input to a question in the current slide. *submitted*: learners had submitted an answer to the questions of the current slide. *idle*: users had more than 10 seconds of inactivity during the current slide.

The results from the input and submit conditions probed us to investigate the temporal dimension of question slides. Figure 7.2 shows a temporal view of all the combinations of gaze and visiblity for the lectures *Prolog01b*. We also included NULL events, which were not included in our table analysis, for completeness. Vertical blue lines denote a transition to a slide that contains questions and gray transitions to slides that do not. The light blue overlay indicates time passing during slides that contains questions. It is noteworthy that when a new slide with questions comes up we have picks of the fixation on-screen/visible combination which after a while is substitued for the fixation off-screen/visible combination. We notice similar patterns in the rest of the lectures in Figure 7.3, Figure 7.4 and Figure 7.5.

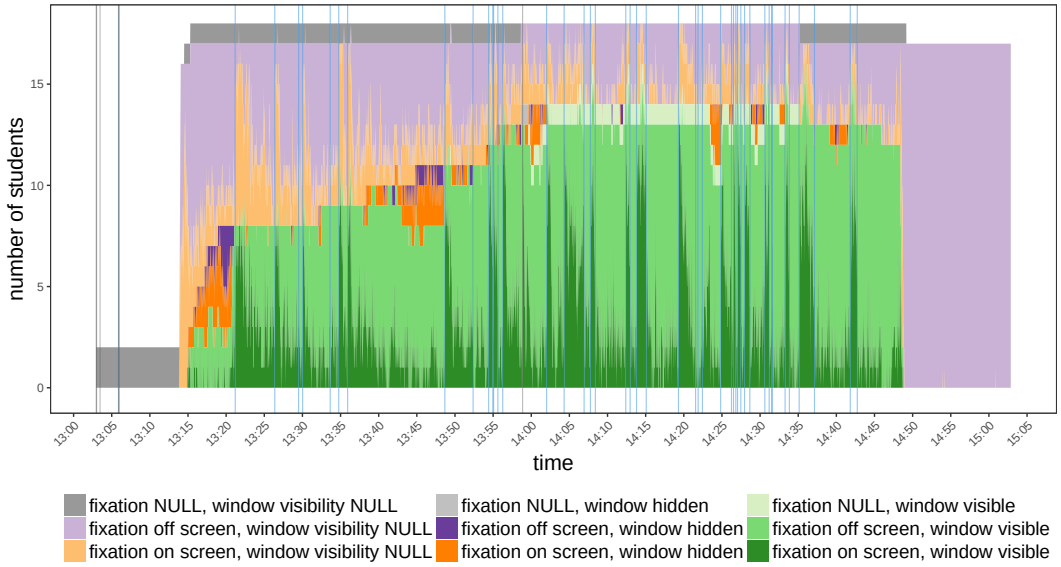


Figure 7.2. State combination of fixation on/off screen and visible/hidden `ASQ` window for all students of the lecture *Prolog01b*. **NULL** values are displayed for completeness.

7.3 Discussion

Going back to *RQ1.4* our exploratory results show in general the teacher cannot be sure were the visual attention of a student is directed at simply by looking at the visible activity indicator. This changes only when the student performs input activity in a question, in which case it very like that the student is gazing at its screen. However, the eye-tracking data indicate – similar to all our case studies – that during slides that contain questions, the attention to on-task activities (i.e., looking at the `ASQ` window) increases. This experiment had two shortcomings that can be easily tackled: a low number of sessions and a low ratio of slides without questions versus slides with questions. An experiment with more lectures sessions and a higher ratio of slides without questions versus slides with questions can give us more confidence in our results and better profiling of visual attention bewteen the two formats. Finally, It would also be of great interest to us to look into how the answer depth format affects student attention.

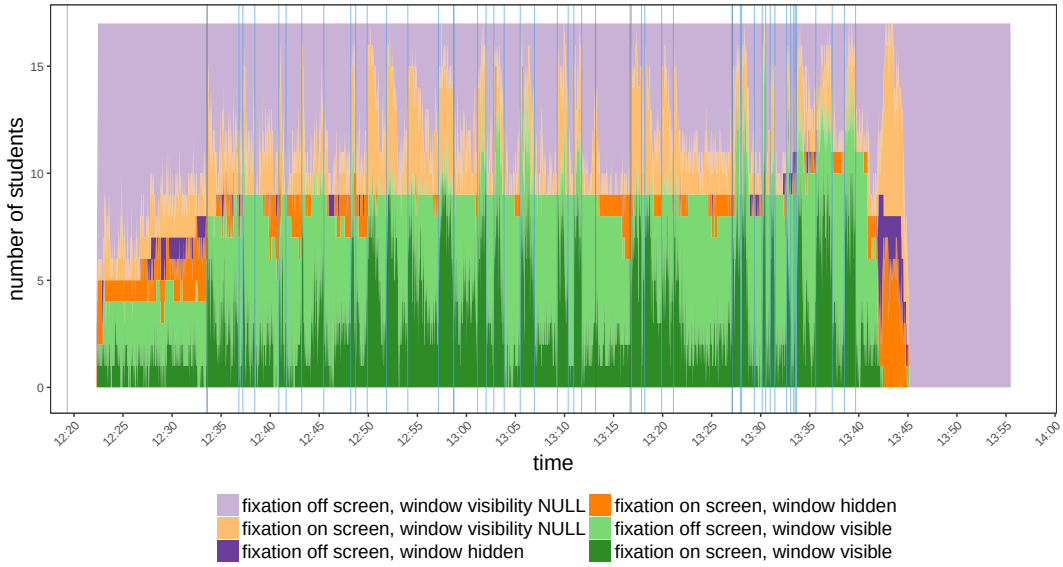


Figure 7.3. State combination of fixation on/off screen and visible/hidden `ASQ` window for all students of the lecture *Prolog02a*. **NULL** values are displayed for completeness.

7.4 Summary

While this has been exploratory work in an experiment with much more data to examine and directions to research, some clear patterns during practice question slides emerged. It seems that the visual attention of students is associated with the task at hand and during non exercise slides their gaze wonders of screen. It remains an open challenge to identify what exactly their visual attention is focused at and look forward to working on it.

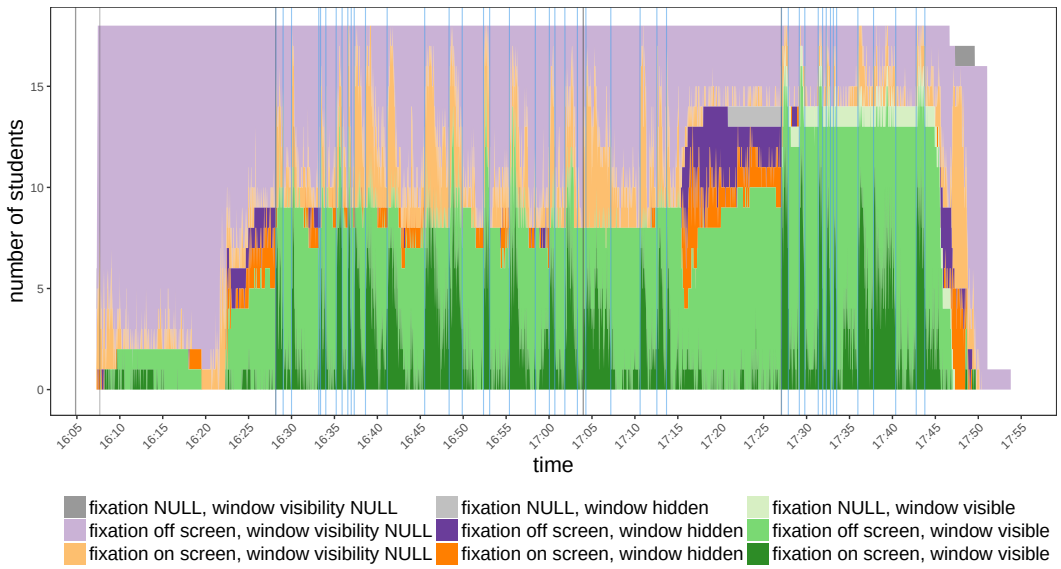


Figure 7.4. State combination of fixation on/off screen and visible/hidden `ASQ` window for all students of the lecture *Prolog02b*. **NULL** values are displayed for completeness.

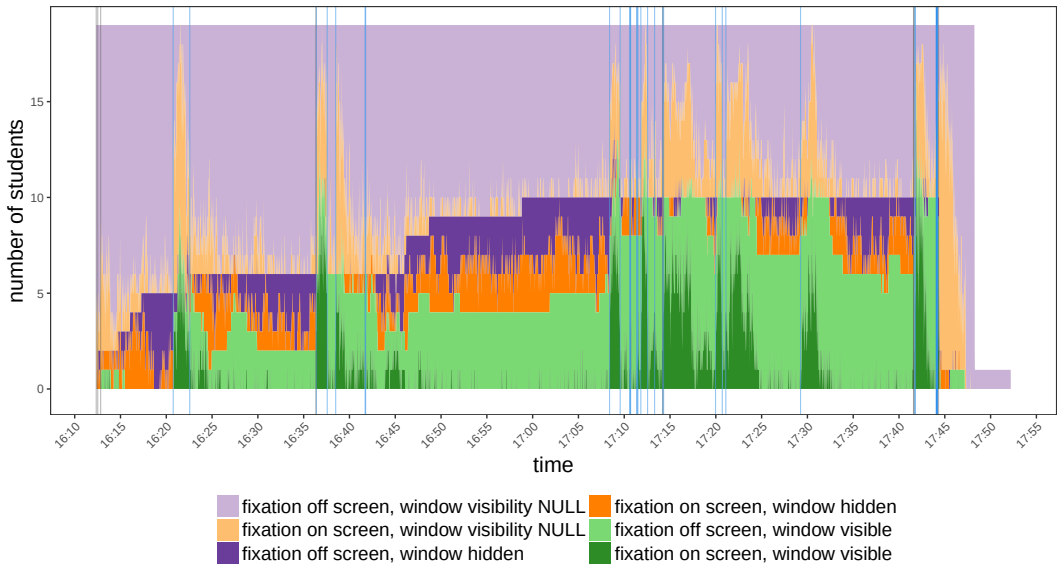


Figure 7.5. State combination of fixation on/off screen and visible/hidden `ASQ` window for all students of the lecture *Prolog03a*. **NULL** values are displayed for completeness.

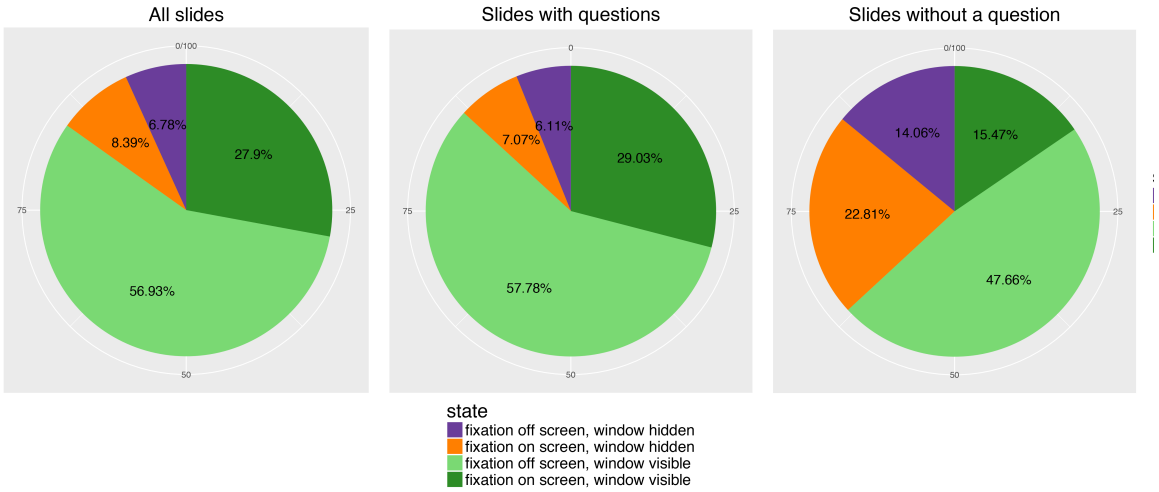


Figure 7.6. Pie charts with all the combinations of fixation on/off screen and visible/hidden `ASQ` window. Each pie chart contains the mean for all lectures.

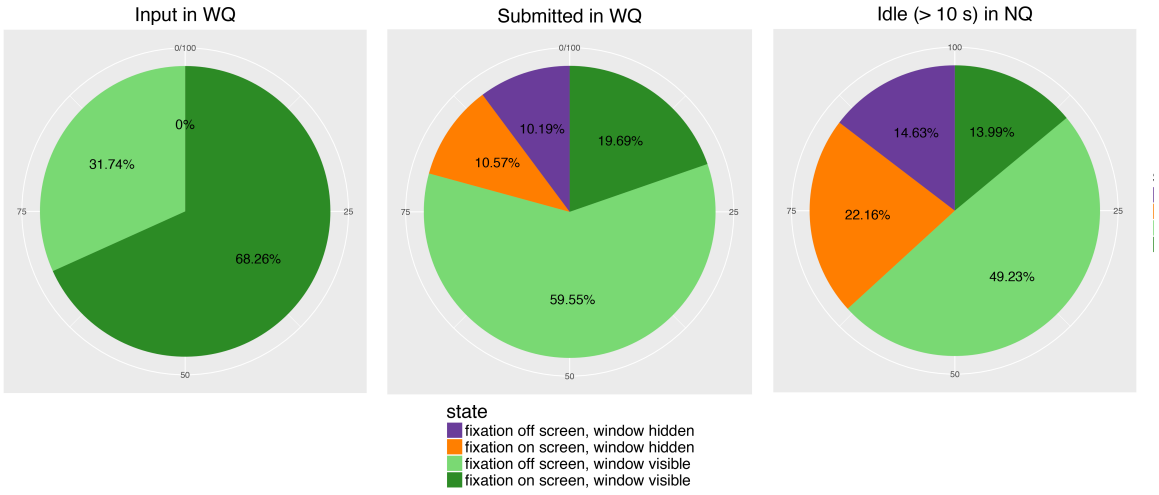


Figure 7.7. Pie charts with all the combinations of fixation on/off screen and visible/hidden `ASQ` window for various *conditions*. Each pie chart contains the mean for all lectures. **Legend**. *WQ*: Slides that contained questions. *NQ*: Slides without a question.

Chapter 8

Case Study: Practice Question Strategy

One dimension of practice questions that so far has not received a lot of attention is their spacing in time as they are interleaved with slides during the entire class duration. As we discussed in subsection 2.2.2 to what extent the placement strategy (e.g. whether interspersed or at the very end of a lecture) of practice questions within a lecture influences retention is still a relatively open question [182]. Next to exploring the impact of ASQ on several student behaviour metrics in section 5.1 and chapter 6, in this chapter we present a case study where we investigate whether different in-class question spacing strategies impact student learning and engagement. This case study is based on the logs collected in the same longitudinal study presented in section 5.1.2 and aims to answer the following research question:

RQ1.5 *Does the practice question strategy — questions deployed uniformly across the entire lecture or with bursts of several questions at the same time – has any impact on student engagement and student learning?*

8.1 Methodology & Use Case

As noted in subsection 2.2.2 there is little research (apart from [182]) exploring the advantages or disadvantages of certain question spacing strategies in the classroom. To fill this gap and inspired by [182] the course instructors initially designed the set of questions in each lecture according to one of the following three question strategies (with the strategies being randomly assigned to each lecture):

Burst (b): questions appear in bursts, each burst contains at least two questions; bursts can be randomly spread across a 90 minute lecture.

Uniform (u) questions are distributed uniformly in time across a 90 minute lecture;

Increasing (i) questions are placed with increasing frequency towards the end of each 45 minute period (two such periods exist per lecture with a 15 minute break in-between).

Figure 8.1 shows three example lectures (i.e. Cookies & session, SQL continued, Database introduction) and their question distribution across time.

Table 8.1. Lecture overview, reported in temporal order of delivery. These are the same lectures summarized in Table 8.1 with a focus on metrics pertinent to this study. **Legend**. **INS**: instructor ($I1$ or $I2$). **QS**: the question strategy (b=burst, i=increasing and u=uniform). **#EQ** and **#HQ**: number of easy and hard questions in the slides. **%SQ**: the percentage of students connected to `ASQ` and answering at least one question (≥1 question submitted). **%AQ**: percentage of students connected to `ASQ` and submitting an answer to all questions. $\rho(S,Q)$: Spearman rank order correlation between slide and question engagement scores. **%CA**: percentage of correct answers. Correlations significant at the $p < 0.01$ level are marked with a †.

Lecture	INS	QS	#EQ	#HQ	%SQ	%AQ	$\rho(S,Q)$	%CA
HTTP	I1	b	6	4	74.5%	26.1%	0.72†	62.16
HTML	I1	u	4	4	79.1%	27.6%	0.58†	47.75
JavaScript	I1	i	5	5	81.1%	8.0%	0.69†	48.14
Database introduction	I2	i	6	1	78.9%	10.5%	0.60†	49.66
node.js	I1	b	5	3	74.1%	15.6%	0.66†	65.53
CSS	I1	u	2	5	79.3%	21.5%	0.72†	64.88
SQL introduction	I2	b	4	5	79.6%	8.2%	0.57†	52.05
SQL continued	I2	u	6	2	76.8%	2.9%	0.44†	54.41
Cookies & session	I1	b	5	4	78.3%	11.6%	0.51†	49.76
Advanced SQL	I2	u	7	3	62.8%	0.0%	0.64†	65.52
Web security	I1	u	4	4	74.8%	14.0%	0.50†	8.46
Conceptual design	I2	b	11	2	83.3%	13.0%	0.42†	60.26
ER logical design	I2	u	4	4	84.7%	2.4%	0.40†	54.49
NoSQL	I2	b	8	0	69.1%	11.8%	0.39†	61.61

As a reminder, the setting of this case study was the 10 week real-world teaching Web technology and database concepts that was presented in section 5.1.2. Table 8.1 displays a summary of the lectures. Overall, six lectures implemented the bursts question strategy and six lectures implemented the uniform question strategy, equally distributed across both instructors. Every lecture contained between seven and 13 question of various types: multiple-choice, multiple-answer, highlighting, SQL-query creation and fill-in-the-blank questions; in all but one lecture (CSS) the easy exercises outnumbered the difficult ones.

After both class instructors had trialed all three question strategies in the first six lectures, it was decided to drop the increasing questions strategy from further consideration as it turned out to be very challenging to align the lecture material with this strategy. Subsequently, in the remaining 8 lectures only the burst and uniform question strategies were implemented.

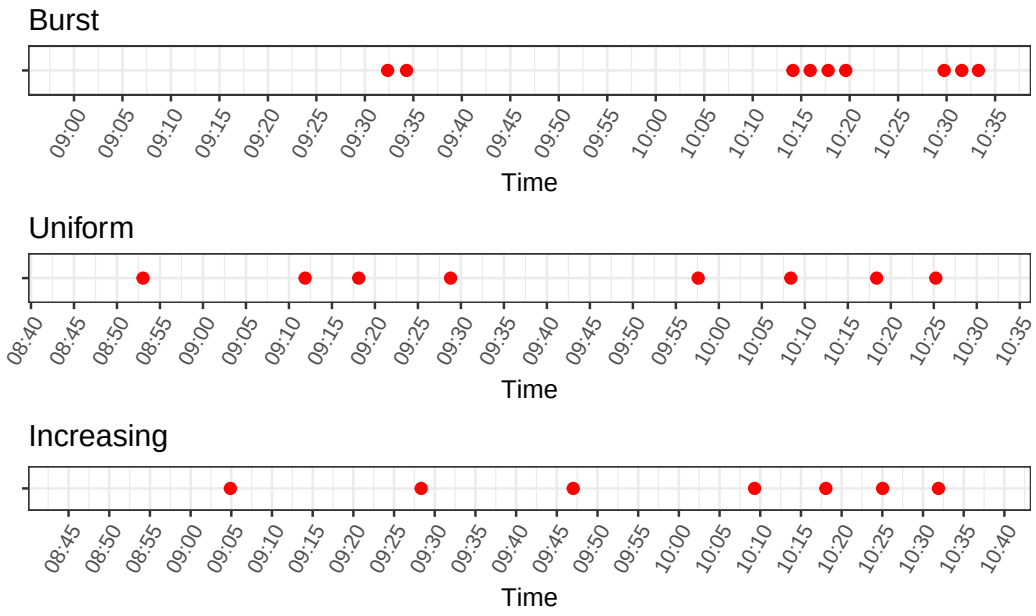


Figure 8.1. A temporal view of question strategies - each dot represents when a question was asked.

8.1.1 Question Difficulty

In our analysis, we also consider the difficulty of the questions posed to the students. All questions were created by the instructors of the course and manually annotated in accordance with the revised Bloom's taxonomy [10] — questions on

Table 8.2. Overview of correlations between slide and question engagement, computed separately for each instructor, each question strategy and the four combinations of instructor and strategy. N is the number of items to correlate. %CA is the percentage of correct answers. Correlations significant at the $p < 0.01$ level are marked with a †

Condition	N	$\rho(S,Q)$	**%CA**
Question strat. *uniform*	768	0.50†	51.92
Question strat. *bursts*	851	0.61†	59.65
Instructor $I1$	1224	0.63†	55.47
Instructor $I2$	741	0.45†	55.02
$I1$ +*uniform*	467	0.58†	50.25
$I1$ + *bursts*	582	0.67†	60.85
$I2$ + *uniform*	301	0.39†	55.58
$I2$ + *bursts*	269	0.47†	56.75

the *remembering* and *understanding* level were considered to be *easy* and questions on the *applying* level were considered to be *difficult*. None of the questions belonged to higher Bloom levels.

8.2 Results

Are questions strategies influencing the engagement levels of the students? To address **RQ1.5**, we analyse slide and question engagement for lectures adopting *uniform* and *burst* question strategies. The results are reported in Table 8.2 (rows 2-3).

The correlation between slide and question engagement scores is higher for the *burst* than the *uniform* condition. This means that in the *uniform* condition with one question on the topic of the previous 5-10 slides, students are more likely to engage with the question than in the *burst* condition where at least some of the questions are likely to be about material more than 10 slides in the past. Drilling down further, we also consider in the analysis the role that the instructors, with their individual lecture style (and question styles) as well as addressed topics, might have played on the engagement level of students. Table 8.2 (rows 3-8) show that instructors clearly played a role. Despite the absolute differences in correlations, we observe that for both instructors the *burst* strategy leads to a higher correlation between slide and question engagement than the *uniform* strategy.

8.3 Summary

In this case study, which was part of a largest experiment to gain a better understanding on the *positive* role that technology can play in large higher-education classrooms, we show show that question strategies can be of influence, but also observe the important role played by lecturers and lecture topics.

It is worth mentioning that the design of lecture material for a system like ASQ can be challenging for the instructor. In addition to obvious technical complexities (such as the creation of slides in HTML5), the design of a lecture experience enhanced with several, pertinent, and rich questions is not trivial. While this is an intrinsic difficulty in the design of lectures, we felt the problem to be exacerbated by the need for pre-defined question strategies, which sometimes "forced" the redistribution of content to account for the temporal allocation of questions.

This chapter concludes the main body of work of this dissertation. In the final chapter we will present conclusions from this work and its main contributions and limitations. We will finish by reflecting on the implications of our work and discuss the next steps.

Chapter 9

Conclusions

In this dissertation we examined how technology can help scale active learning to- and capture the attention of large student cohorts which are becoming more common in post-secondary education. Bachelor-level courses are lectured in large classrooms that can easily include hundreds of students. Under such conditions, students are easily distracted — especially through the ubiquitous availability of digital devices. Our modus operandi is "technology as an aid to pedagogy". To gain a better understanding on the *positive* role that technology can play in large higher-education classrooms we developed a privacy-preserving teaching tool that turns students devices to affordances of learning. We conducted usability studies with students to verify the usability of the system and make sure that students would be willing to spend their time following the course instead of engaging in off-task behavior. One big design factor in our implementation that we think can help scale active learning, was tackling teacher bias by attempting to aggregate all answers and, whenever possible, group submissions to provide a compact form of the level of understanding in the classroom. This also helps in maintaining instructor workload within acceptable levels which is evidenced from positive workload-level ratings from teachers. We reported on case studies where our tool has been put into use to support lectures with interactive teaching elements and explored how to scale active learning in large classrooms. Overall, our findings demonstrated the suitability of `ASQ` as platform for "in-situ" privacy-preserving research in education. Notably, we confirmed once more the positive impact of practise questions in sparking the interest of students on taught material. We also unveiled the potential of such a tool to increase teacher awareness of the student behavior via analytics mined from student devices. We compared activity in passive and active segments of a lecture to find that increased passive engagement has a positive correlation with active

engagement. While this can be a good predictor for increased performance in a course, it does not answer how to make passive segments more engaging. To this end, we looked into the variation of attention between segments and confirmed findings that demonstrate a visible impact of introducing interactive elements on the collective attention of the entire class. We also investigated how different practice question strategies affected students' engagement with lecture material. We showed that question strategies can be of influence, but also observe the important role played by lecturers and lecture topics. Instructors can utilize these analytics during a lecture to adapt their learning on the fly or consult them in designing future editions of a lecture to better address the needs of students. In the reminder of this chapter, will first summarize the answers to our RQs stated in chapter 1 and then provide a brief summary of the contributions and limitations of our work. Finally, we will look into the implications of our work for technology-enhanced education and our vision of the modern classroom and discuss future steps towards accomplishing it.

9.1 Summary of Contributions

We investigated to what extent can teaching computer science in modern brick-and-mortar classrooms benefit from introducing technology that 1) facilitates active learning by interleaving slides with active learning question types and 2) provides data analytics capabilities to give insights on the level of understanding of the audience and its behavior dynamics. To explore our research questions (RQs) we introduced `ASQ`, a Web platform for interactive presentations that supports complex question types and real-time analytics. We reflected on the design and implementation of such a platform and our experiences from using active learning question types with students. In a series of a case studies in real-world BSc and MSc Informatics courses we managed to observe a positive shift to using student devices as affordances of learning as opposed to distractions. Our results demonstrate how engagement increases when active learning questions, an established method associated with better learning outcomes, are embedded into lecture slides. In our studies we also utilized the data analytics capabilities of `ASQ` to infer student attention and results suggested that we are in line with the findings of the literature on the subject. Finally we set to discover whether the insights on behavior dynamics obtained from `ASQ` match the actual behavior of students.

9.1.1 Answers to Research Questions

In section 1.2 of chapter 1 we set as the main goal of this thesis to look for an answer the following research question:

RQ1 *To what extent can a Web-based, privacy-preserving teaching tool be reliably used to enable active learning, infer student behavior dynamics, assess the attention and learning outcomes of students and provide insights to improve teaching?*

As this is a RQ that covers a broad range of topics in education technology, we focused our work in providing answers to smaller problems that were formulated by refining the RQ into the following:

RQ1.1 *To what extent can a Web-based, privacy-preserving teaching tool be reliably used to assess student behavior dynamics and learning outcomes of students from their interactions with the tool?*

In chapter 5 we unveiled the potential of browser event-driven analytics in providing engagement metrics for the students' behavior during two types of slides: those that contained lecture material and those that contained questions. We confirmed prior findings that report student distraction from their devices and saw how ASQ mitigates this by engaging students through questions. Furthermore, we observed a clear correlation between slide engagement and question engagement: students that engage more in following lecture slides tend to answer more questions. Finally, feedback from students on the usability and imposed workload respectively was overly positive showing that introducing ASQ in the classroom adds an easy to tolerate overhead. In our opinion, these results render ASQ as a suitable tool for teaching and experimenting in real-world courses.

RQ1.2 *To what extent can students' attention be inferred from their interactions with a Web-based application?* Drawing on the browser event concepts of the previous *RQ1.1*, in chapter 6 we showed how to infer high level attention states for the audience of the classroom. Based on the events logged in ASQ, we are able to infer in real-time (and live in the classroom) when and to what extent the attention fluctuates over time. Our findings were, in general, aligned with the literature and make an interesting case for using ASQ in a lecture: Students' attention drops over the class period but rises when when interactive elements (questions) are introduced. Also immediately after interactive teaching elements, the level of distraction is lower than before the start of the interaction.

RQ1.3 *Which type of interactions are most correlated with (in)attention?* In chapter 5 we observed that not engaging with slides that contain lecture mate-

rial can lead to less engagement when answering questions. In chapter chapter 6 we quantified this more defining inattention-oriented activities/states and attention-oriented activities/states and then examining the correlation each group had with time. The first were shown, as postulated, to be negatively correlated with time while the latter positive. These insights can help instructors adapt their lectures on the fly by monitoring the states of the audience or design their future lectures better based on past data.

RQ1.4 *How does student gaze behavior as recorded through the use of eye-tracking relates to activity indicators generated from browser events?* We seeked answers to this question in the exploratory study of chapter 7 where found that the visual activity indicator of the `ASQ` window is not in general a good predictor of the visual attention of a student. An exception occurs during times that students perform input activity in a question; in those cases the visual attention on the screen is significantly higher. Moreover, we observed patterns in the percentages of visual attention towards the screen and the task at hand: just following the lecture slides (no questions) gathered the least amount of visual attention.

RQ1.5 *Does the practice question strategy — questions deployed uniformly across the entire lecture or with bursts of several questions at the same time – have any impact on student engagement and student learning?* `ASQ`'s data analytics capabilities that helped in answering the previous RQs also played a great role in answering this RQ in chapter 8. We investigated the effect of different strategies of spacing practice questions during a lecture. Our analysis identified the *burst* (questions appear in bursts, randomly spread across a lecture) condition as superior to the *uniform* (questions are distributed uniformly in time across a lecture) in terms of increased correlation between slide engagement and question engagement. Taking also into consideration that specific topics and instructors' teaching style differences affect student engagement, our results can help content creators to design effective question placement strategies in lectures.

9.1.2 A Novel Paradigm to Interleave Study Material Slides with Questions

In chapter 3, we introduced a paradigm for delivering lectures with slides that contain active learning questions. We showed how the Web can facilitate the utilization of student's devices as an affordance to learning by allowing students to follow slides and answer questions embedded within them. This paradigm enables real-time collection of student's submissions, zero context switch between lecture and question answering software and fast feedback-cycles.

9.1.3 A Web Component library of Interactive Question Widgets

We presented our attempt at simplifying the authoring of questions by using Custom Elements to encapsulate the logic and design of full-fledged interactive widgets (exercises, questions, aggregated statistics visualizations, and more) under a single HTML tag.

9.1.4 Using a Custom URL Schema to Reuse Questions into Presentations

In Appendix B, we present a way to inject questions in presentations using a simple URL schema. Questions published on the Web can be shared and reused between instructors just by copying a simple URL into a presentation slide. Our hope is for this mechanism to extend to more educational content (lectures, rubrics, reference answers) and drive educational content sharing and reuse between instructors.

9.1.5 Design and Implementation of a Plugin System to Facilitate creation of Active Learning Question Types

Creative active learning question types for computer science question that can be used within lectures can be a daunting task due to the complex requirements of such an environment. In chapter 4 we outlined the requirements for creating a plugin system to tackle the challenging task of creating plugins for state of the art Web applications that span across both client and server. We presented our reference implementation of *asqium*, a plugin system that follows best practices in software architecture and design to enable the creation plugins that have the following desired properties: multiple decoupled ways to communicate with the core business logic of the application; single repository packaging; easy dependency management; single programming language (i.e., JavaScript) for both server- and client-side; persistence flexibility; asynchronous non-blocking code execution; code and style encapsulation; and theming. We also provided an example on how to create a real complex question type.

9.1.6 Design and Implementation of 14 Question Types to practice Computer Science

Guided by our goal to support support active learning in traditional brick-and-mortar classrooms and based on the aforementioned *asqium* plugin, we imple-

mented 14 question types that were presented in section A.4. 11 of them support cued/free recall: *CSS Select*, *Live JS*, *JS function*, *SQLlite HTML Fiddle*, *Text*, *Code*, *Highlight*, *Rate*, *Java*, *Diagram*. The first five support live programming, a feature that gives students immediate feedback on their efforts inside the browser. The other 2 are recognition question types: *Multiple choice*, *Classify* and *Order*. A lot of them feature analytics capabilities that aim to speed up assessment and accelerate the feedback cycle. Next, to help future question type designers create their own question types, we presented the motivation behind their creation; implementation guidelines; feedback and evaluation from subjects that used it in real-world settings.

9.1.7 Mining Browser Events to infer Higher level Engagement States and Offer Insights on Student Behavior Dynamics

In chapter 5, we studied whether it was possible to utilize browser events such as input, focus or visible to infer student engagement with slides. We presented models for student engagement on study material slides and slides that contained question and we reported a positive correlation between the types of engagement.

Drawing on these findings, in chapter 5 we explored ways to infer student attention from browser events. We resorted to construct high level states that reflect the students attention based on this events and compared our findings to the literature in student attention. Our results aligned with those of the literature, suggesting that `ASQ` can provide an inexpensive and unobtrusive alternative to complex, elaborate or intrusive equipment (like note taking or Electroencephalography (EEG)) that is necessary to gauge attention. We also confirmed that interactive elements like questions drive up engagement and contribute to high levels of engagement even after the end of the interaction.

9.1.8 Insights Towards Making Lectures More Engaging

The insights gathered from the data analytics of `ASQ` can help education content creators and instructors to design more engaging lectures. First by consulting the attention state of the audience, interested parties can identify lecture segments that suffer from low engagement. One of the ways to mitigate low engagement, as discussed in chapter 6, is to introduce a question. Another way is to choose a temporal placement strategy for questions that is optimal for the topic and instructor learning style of the lecture. We observed that instructors are the most

influence factor that could explain differences across lecturers, although *bursts* strategies generally obtain higher slide engagement.

9.2 Limitations

A significant part of our work is dedicated in inferring student attention dynamics. In chapter 5 we presented a case study focused in measuring engagement based on Web-telemetry data. Despite the correlation between active and passive engagement which matched our expectations, it is practically impossible to know exactly what are the actual thoughts of the students and if they are in reality engaged or not. This holds true for the case study in student attention as well (chapter 6). The high level attention states are just intended to inform teachers on how students interact with the platform and in no way represent the actual neuropsychological states of the students.

Another consideration is related to student engagement during question types. The novelty effect may explain part of the effect size when introducing interactive elements in presentations. Due to increased interesting in the new technology (`ASQ`) students may have exhibited higher engagement rates.

9.3 Future Work

We believe that we have just scratched the surface regarding what becomes possible when introducing interactive presentations supported with analytics capabilities within traditional classrooms. Our results and the initial experience of our users provide plenty of inspiration for future work. `ASQ` will allow us to easily perform new case studies in other education settings. The first steps towards a better system would take the feedback we received during the evaluation of the system and its question types into consideration. The current version of `ASQ` is only a first step towards a highly sensor- and data-driven classroom. We envision a classroom where technology monitors not only learners but instructors as well to alert them for alarming levels of cognitive load or erroneous actions. Moreover, instructor decisions during the lecture can be recorded and mined so that they can automatically be predicted, recommended and applied in future lectures. We also see the potential for utilizing the technology we developed in settings out of its original context, for example to post in-lecture answers for post-lecture discussion, share educational resources and past data or for use in MOOCs. In the following sections, we briefly discuss the aforementioned parts

of our vision.

9.3.1 Incremental Improvements

Here we list a number of general, easy-to-implement, small incremental improvements as the first steps to go beyond our existing solution.

During our evaluation phase we gathered a lot of constructive feedback from both instructors and learners on how to improve the live lecture experience: introduce hints for questions; enable student navigable slides so that students can go back and forth to revisit material at will; lock submissions for a question time during the feedback phase and help instructors filter and block spamming in submissions.

Another area we want to focus is content authoring. We wish to accelerate content creation cycles through the implementation of a GUI editor [240; 242] on top of our widget elements that integrates the slide and question creation process. The editor will be integrated with the system to ease uploading and reediting of presentations. Moreover, we want to lift the current predefined static content limitation of `ASQ` by implementing functionality to quickly create and pose questions in the classroom on the fly during a lecture [241].

We also wish to introduce more interactive question types that focus on visualizing algorithms and plots. A powerful design principle is *live tuning* which refers to modifying constant values of interactive applications that modify their behavior [108]. For example modifying the GUI sliders of a software sound mixer affects the resulting sound, or modifying the parameters of a plot affects the graph result. We consider this to be a very intuitive way that can potentially lead to deep learning of complex topics.

9.3.2 Monitoring Instructors

There is the need for more research in measuring and monitoring the cognitive load of the teacher so that it can be relieved. For completeness, in the chapter 2 in subsection 2.2.4 we mentioned three effects of cognitive load: automaticity which can introduce errors under certain circumstances, cognitive defaults and ironic processes. It is thus important to monitor the cognitive load of instructors and alert them when it reaches alarming levels leading to erroneous actions. Most methods to measure cognitive load directly or indirectly (brain activity, heart rate, performing a secondary task, filling in self-report questionnaires, etc.) can hinder the performance of a complex task such as teaching.

Prieto et al. [140] suggested using the eye-tracking in multi-tabletop space setting as an unintrusive and increasingly affordable way to measure cognitive load based on pupil diameter and eye movement data. We believe that this approach could work with a system like ours during the *answer* and *feedback* cycle where teachers usually inspect and assess solutions in front of a monitor. The system could then warn teacher of high cognitive level load and suggest countermeasures such as extending the answering time or moving on with the lecture.

9.3.3 Outside The Classroom

In terms of where learning is happening, the educational landscape can be segmented on three criteria (Figure 9.1):

1. *Online/Offline*. Whether the teaching material is delivered through the Internet or not.

2. *Co-located/Remote*. This criterion refers to whether the participants are in the same physical space or whether some or all are in different physical spaces.

3. *Synchronous/Asynchronous*. Are all participants learning at the same time from an instructor, or are the progressing on their own without an instructor?

Our work focused on the online, co-located synchronous scenario, however our technology can adopted/improved to work in other scenarios as well. For example it can help during the "Homework" phase by publishing student submissions or that were aggregated during classroom lectures or lecture slides in online learner communities (such as Piazza [pia] or Askalot [161]) or Learning Management Systems (LSM) (for example Moodle [61]) to foster post-classroom discussions and knowledge sharing. Members should be able to use the question types of `ASQ` instead of simple text and non-interactive graphics.

The sharing of educational content can also expand to any educational consumer via the `ASQ` URL schema (which should be extended to support individual slides, rubrics and answers). `ASQ` was in part inspired by MOOCs and we are confident it can be used in such scenarios where students will be able to engage in active learning using the same question types developed for the in-class version (an online, remote and asynchronous scenario). A benefit of using it in such a setting is the lack of time and cognitive load constraints to instructors imposed by the classroom environment. This is however outweighed by the need to grade potentially thousands of open-ended questions, a challenge we discuss in subsection 9.3.4.

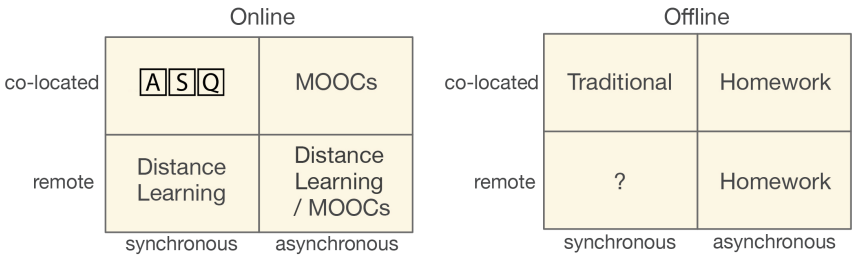


Figure 9.1. Where learning is happening

9.3.4 Submission Clustering

Research shows that "typically teachers in traditional classrooms give students in the top third the greatest attention and students in the bottom third receive the least attention and support" [28]. Teachers are frequently unaware of the fact that they are providing more favorable conditions of learning for some students than they are for other students. Generally, they are under the impression that all students in their classes are given equality of opportunity for learning. This probably gets worse in bigger classrooms. Together with the cognitive load limitations we thoroughly discussed, we think it is worthwhile to utilize technology and analytics to provide teachers with a "big picture" view of the level of understanding in the whole classroom by clustering responses to quizzes.

In the computer science domain, our empirical data show that usually the number of individual submissions will be significantly higher than the number of commonly recurring answer patterns in them. We can cluster results based on well defined features to provide the teacher with "meaningful" clusters [190] instead of the time consuming alternative of sifting through each answer. For programming assignments, the most popular way to represent program structure is *Abstract Syntax Trees* (ASTs). It is common to build higher level structures based on ASTs to achieve meaningful comparison like comparing "code phrases" or subgraphs of ASTs [129].

With regards to short text answers, in [95] we used short text submissions collected from `ASQ` as a data source to successfully create clusters of similar solutions in a post-class analysis. We aim to adapt our work for in-class use to facilitate instructors in assessing submissions. Moreover, Shresta et al. [160] demonstrated that "corpus based similarity measures do not significantly affect the clusters and that the performance of spectral clustering is better than hierarchical clustering" in a study where they evaluated corpus based similarity methods (like Cosine similarity, Latent Semantic Analysis, Short text Vector Space Model, and Kullback-Leibler distance) and clustering methods (like Complete

Link, Single Link, Average Link hierarchical clustering and Spectral clustering) with three evaluation methods (clustering F-measure, adjusted Rand Index, and V perform). We would like to see if we can replicate these finding for short text student submissions in a classroom setting. Rangrej et al. [143] showed that graph based methods with affinity propagation performs better compared to K-means and SVD-based methods when applied on short text data collected from Twitter.

9.3.5 Adaptive Teaching and Predicting Analytics

We are very confident that systems like ASQ can also be used to support adaptive teaching. Campbel and Oblinger defined the five steps of academic analytics: capture, report, predict, act, and refine [38]. Our work focused on all but the predict step. We think that it is worthwhile to attempt to further exploit ASQ's attention level monitoring capabilities to recommend teachers subject-related questions that could be used to restore focus, if an attention drop is detected during the presentation of slides. Conversely, during exercises, after the majority of students have answered or are idle the system can notify the teacher to move on with the lecture, which can lead to gains in both attention levels and lecture time. For future work we think it is interesting to focus on the following questions: (i) how different instructors influence student learning and student engagement and what are the causes for the observed differences, (ii) how more complex and time-consuming question types (e.g. to implement an entire program) can be broken down and incorporated into the lecture, and (iii) how the real-time insights we have about students' engagement can be reflected back to them in the live classroom to raise awareness and self-reflection.

Finally, we believe that our technology can augmented to automatically drive some of the orchestration activities in the classroom by relying on past data of instructor decisions to predict future ones. Such decisions include grading questions, providing hints when students are stuck at a particular step of a solution or in a particular way, notable submissions that warranted more discussion and more.

9.4 Final Remarks

Our work focused primarily on post-secondary lectures with a large number of participants. Our results can find their way into the classrooms of disciplines with similar requirements for lectures that include active learning segments and pro-

mote timely interactions and feedback between the instructor and the learners. The first examples that come to mind STEM fields or the heavily PBL environment of medical studies. Indeed, our approach to inferring student behavior, such as attention, needs little to no adjustments to be adopted in these contexts. Moreover our design for scalable active learning practice question types paves the way for more tools that are optimized for active learning in other specialized domains. The implications of our work can also extend to other educational or professional contexts with a need for active audience engagement such as K-12 education and technical training of employers.

Our data-driven approach and utilization of the Web to scale one pedagogy in large audiences can be complemented with research in learning retention and data from additional sensors and technologies (eye-tracking, heart sensors, skin conductance and activity sensors) as advancements in technology lower the barriers in adopting them. We hope in this fashion to obtain more accurate models on how student learn and a more accurate picture of the physiological, mental and emotional state of the audience respectively. These insights can allow instructors to adapt their teaching at very fine-grained level while making the most of their time in the classroom.

Appendix A

ASQ User Guide

A.1 **ASQ** Classroom Cycle

Presenters that wish to introduce ASQ in the classroom need to first prepare their lecture presentation using ASQ's HTML5 presentation format as discussed in detail in section A.2. Authoring such a presentation takes longer than with Graphical User Interface (GUI) presentation editors, due to the markup involved and the overhead of embedding questions in the presentation. The markup format however has the advantage of enabling finer-grained control over the content and styling which allows presentations to act as full fledged applications. After testing the presentation locally, the presenters need to upload the presentation to the ASQ platform and start a "live" session.

At the start of a live lecture session the students may connect to the presentation. They can either follow a direct link, which will usually be projected on the beamer using a specialized ASQ component called `<asq-welcome>`, or visit the "live presentations" page of a presenter. Each presenter account has an associated "live" webpage that lists all the ongoing live presentations. If there is just one live presentation, the user is redirected to the live presentation. Students may bookmark the Uniform Resource Locator (URL) of the live presentations page to easily access subsequent live presentations of a course.

While ASQ can be used in a variety of ways during a lecture, in our courses at Università della Svizzera italiana we use a specific flow that has been influenced by research findings in the fields of attention and retention as well as our experience, called the "ASQ cycle". We use active learning question types to increase the attention of students due to the introduced interactivity [35] and, most importantly, encourage active – cued or free – recall of the taught material [114; 76; 83]. The cycle consists of the following phases:

Figure A.1. The ASQ classroom cycle.

Present The cycle can start or end with the presenter introducing material from a study subject as would happen in a regular lecture. The order in time of this step depends on the teaching style and objectives of the instructor. For example, starting a new topic with a question may help to assess the preconceived notions of the class and motivate the students to listen to the following explanation. Conversely, following a presentation with questions can be used to check if the students have been actually listening and also evaluate their short-term retention and understanding. In any case, it is generally considered a good practice to have short segments of lectures interspersed with other, more interactive activities (in our case practice questions) [82; 29; 32].

Ask In this phase the presenter poses questions to the students to assess their level of comprehension. Question types of free/cued recall and recognition answer depth format are supported, but the first two are preffered since they correlate with better learning outcomes [83; 106; 120].

Answer In this phase the students will try to recall the taught concepts and apply them to answer the posed questions. This process can help them gauge their level of understanding and detect potential gaps in their knowledge or misconceptions. Optimizing the duration of each phase and determining how to interleave them are challenging tasks for teachers adopting ASQ. One of the challenges is an effective strategy the temporal spacing of practice questions during the lecture, a problem we look in detail in the case study presented in chapter 8.

Assess This phase stars with the first student submission. Submissions of all question types can be assessed manually by the presenter. Closed question types can be assessed automatically. Students can also engage in peer/self assessment

to learn by looking at the solutions of others and remain focused on on-task activities [238]. Some open ended question types support clustering as mentioned in section 3.3. In this phase the presenter accumulates information of how well the audience has absorbed and understood the concepts that are test with the current quiz.

Review In the final phase of the ASQ cycle, students receive feedback from the presenter about their submissions, which further helps to cover gaps and clear misconceptions. This phase also influences the evolution of the rest of the lecture, for example teachers might need to repeat concepts or skip topics completely.

A.2 **ASQ** Authoring Workflow

In this section we will describe the workflow that content authors – usually instructors of a course – who want to use ASQ presentations may follow. We will cover slide creation, adding ASQ widgets like question types, testing and starting a live presentation.

ASQ presentations comprise two main parts: 1. a main *HTML* file that contains the slides, *ASQ widgets* and references to external static dependencies such as libraries, stylesheets, images, vector graphics and videos; and 2. the rest of the static assets. Content creators can choose between two JavaScript presentation frameworks for HTML5 presentations: *impress.js [Szo11]* and *reveal.js [EH11]*. Both represent slides using HTML5 markup. In impress.js slides are HTML5 elements that have a `step` class attribute while in reveal.js `<section>` elements. Listing A.1 shows an example slide for the former while Listing A.2 shows a slide with the same content for the latter. Notice that impress.js supports 3D position of the slides in an infinite canvas but requires the author to position each slide manually. reveal.js on the other hand automatically layouts the slides in a 2D space. The resulting rendered slides are shown in Figure A.2 and Figure A.3 respectively (with different CSS styles).

```
1  <div class="step" data-x="1000" data-y="500" data-z="100">
2    <!-- Slide contents go here -->
3    <h1>Slide title</h1>
4    <img src="img/slide_image.png" alt="The main slide image">
5    <ul>
6      <li> item 1</li>
7      <li> item 2</li>
8      <li> item 3</li>
9    </ul>
10 </div>
```

Listing A.1. Example slide of an impress.js presentation

```
<section>
  <!-- Slide contents go here -->
  <h1>Slide title</h1>
  <img src="img/slide_image.png" alt="The main slide image">
  <ul>
    <li> item 1</li>
    <li> item 2</li>
    <li> item 3</li>
  </ul>
</section>
```

Listing A.2. Example slide of a reveal.js presentation

Figure A.2. Rendered impress.js slide for the code of Listing A.1

Authors can include interactive ASQ widgets anywhere in their presentation – the most common use case is inside slides– using the interactive presentation markup [242] which is explained in depth in the next section. Of course, they can also include any static asset that is supported by HTML5. For the ASQ widgets to function properly, it is *important* to import their definition either by installing them locally through the *bower* front-end dependency management tool or by supplying a URL of the widgets that are hosted in the ASQ server. The

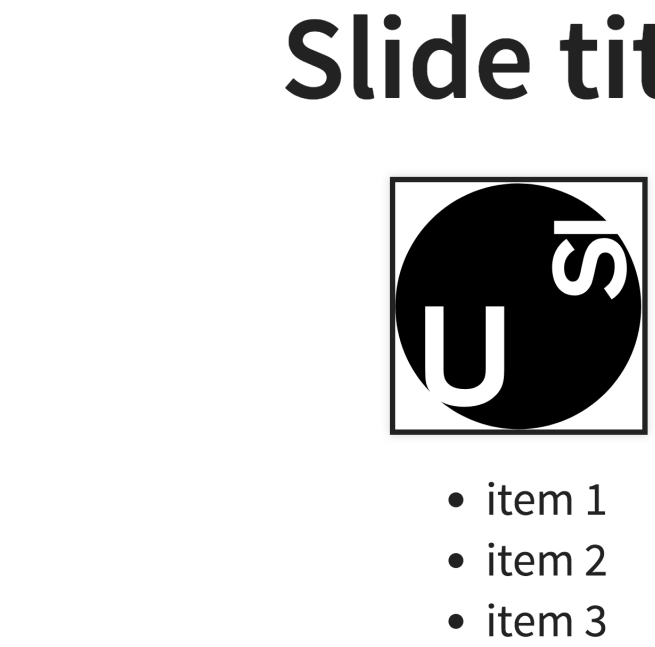


Figure A.3. Rendered reveal.js slide for the code of Listing A.2

definition of the most important elements is packed in *one* HTML import called `asq-elements.html` for convenience.

To test the presentation without the ASQ widgets, it suffices to open the main file of the presentation with a modern browser. To test if ASQ widgets render correctly albeit without being able to interact meaningful with them (for example by submitting answers), authors can serve the main file of the presentation through a simple static Web server.

To test and use the presentation with a live audience, authors need to upload a ZIP archive to an ASQ server. The archive should contain in its root directory the main presentation file and any assets that are referenced with relative URLs from the main file. The upload can be performed by a drag-and-drop GUI visiting the *"upload"* Web page (top of Figure A.4) the corresponding ASQ website or by using the *curl* command-line tool to zip and stream the presentation simultaneously. The latter can be faster especially during the testing cycle. The command to use for uploading with curl is unique for each user; for new presentations it can be obtain in the "upload" Web page (bottom of Figure A.4), while for re-uploading an existing presentation the author has to click the dedicated icon of the target presentation in the *"presentations"* Web page to pop up a window that displays the command (Figure A.5).

Once the presentation has been uploaded, users need to start the presentation

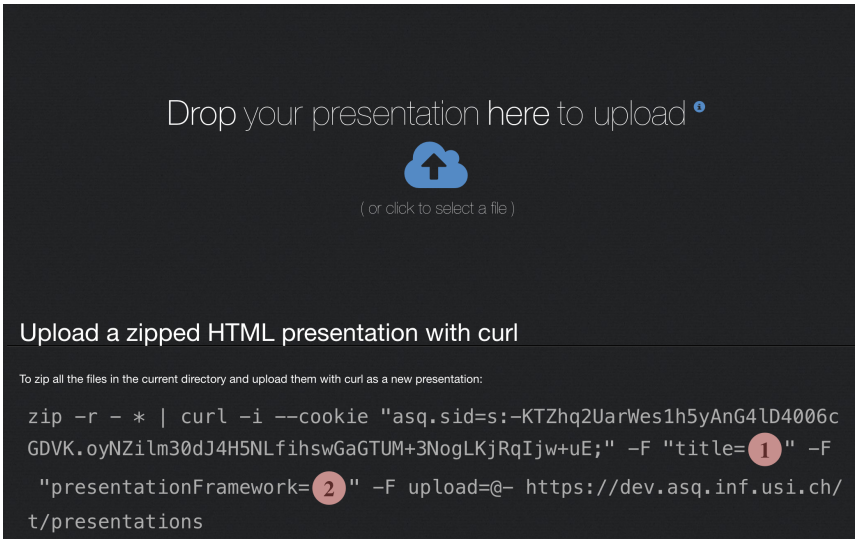


Figure A.4. The "upload" page of ASQ. On the top there is the drag-and-drop area that accepts ZIP archives. On the bottom, the command to upload a new presentation via *curl* is displayed.

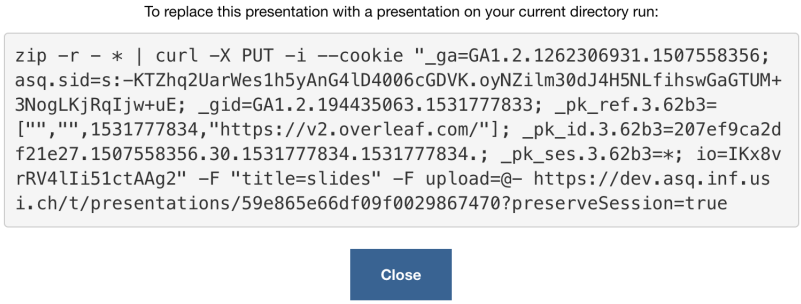


Figure A.5. A pop-up window displaying the reupload command for an existing presentation.

by clicking the *"play"* (left of Figure A.6) button in the "presentations" page. The presentation will go *"live"* and audience members will be able to connect using a URL that can be obtained either by an tf <asq-welcome> widget embedded in the presentation (presented in detail in subsection A.3.2) or by visiting the *"live"* page of the presenter. Live presentations can be stopped by clicking the *"stop"* (left of Figure A.6) button in the "presentations" page. When presenters re-upload a live presentation, the new version is automatically started into "live" mode.

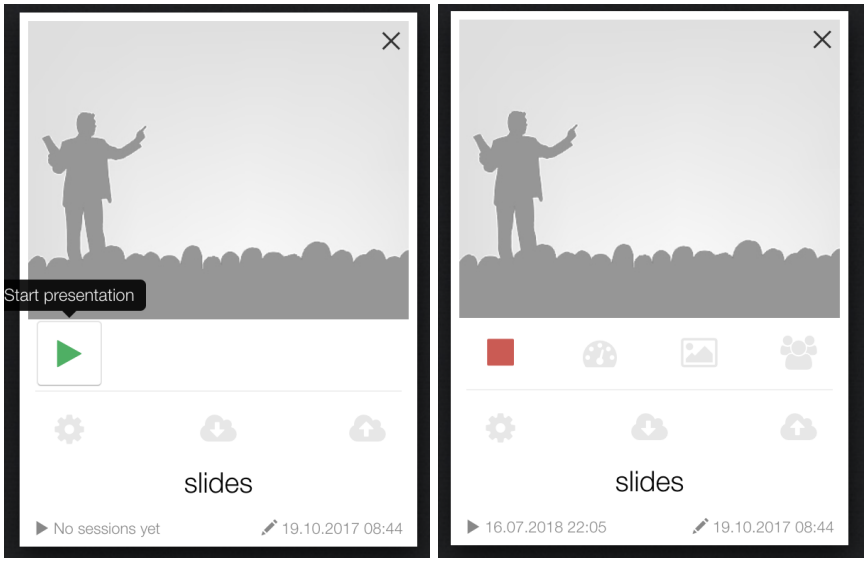


Figure A.6. Left: a presentation in the "presentations" page with the *"play"* button that starts the presentation highlighted (green color).
Right: a presentation that is "live" in the "presentations" page with the *"stop"* button that stops the presentation highlighted (red color)

A.3 Interactive Presentation Markup

Authors have a set of `ASQ` related widgets that they can include in their presentations to make them interactive. `ASQ` widgets are built on top of *asqium*, the plugin system of ASQ that is presented in section 4.2. While each plugin has both frontend and backend parts, in this section we will demonstrate usage from a presentation author's point of view. Thus, we will just display code pertinent to the frontend logic.

The frontend of each widget is implemented using the Web Components [99] technologies [244]. Each widget is a registered Custom Element [60] that can be used the same way a regular HTML Element [1] is. Common functionality that is required within different widgets (usually question types) is abstracted into separate reusable `common` elements.

In `ASQ` each user of a live presentation has a *role*. The two most important roles are:

- **presenter**. This role is associated with the presenter of a live presentation. Allowed actions are advancing the slides, viewing responses and drawing on a presentation.

- **viewer**. Users with the "viewer" role are usually audience members. Their device synchronizes slides with the presenters. They can view and answer questions.

Most of the widgets render different content for different views. For example an exercise rendered for the viewer view will render a "submit" button, whereas the same exercise for a presenter view will display an activity progress bar for the submission process as demonstrated in the top and bottom of Figure A.7 respectively.

The most important widgets are questions of different types that will be discussed in detail in section 3.3. The rest of the widgets are presented in the remainder of this section.

A.3.1 `<asq-exercise>`

Within an `ASQ` presentation, a question should always belong to an *exercise*. An exercise is synonymous to a quiz; which can contain one or more questions. They are implemented using the `<asq-exercise>` element. All children (first level descendants) HTML elements of an `<asq-exercise>` element that are `ASQ` question types are considered to be parts of the quiz. Standalone questions that do not belong to an exercise will be displayed but students will not be able to submit their answers. This setup can be used for practice.

The `<asq-exercise>` will render one "submit" button as shown on the top in Figure A.7 for the viewer view. Clicking the submit button will result in an exercise submission that contains the values of all the question elements of the group at the time of submission. Optionally, authors can set the binary `confidence` attribute to display a 5-star rating widget. Viewers can report how confident they are about the answer they provide by using the 5-star rating widget. The presenter view of the `<asq-exercise>` element displays a progress bar that shows the cumulative viewers' activity states for the corresponding exercise. Each user can be in one of the following activity states:

idle The rightmost partial progress bar displays users that are idle, that is, they have not performed any activity for more than 10 seconds in the `ASQ` window and have not submitted an answer. Defaults to gray.

focused Second from the right is the partial progress bar that displays whose `ASQ` browser tab is in focus, have not submitted an answer and are not idle. Defaults to light blue.

working This partial progress bar displays users that are working on the `ASQ` window and have not submitted an answer. As "work" we consider *input* [mdn18c] or *click* [mdn18b] browser events. Defaults to orange.

submitted The leftmost partial progress bar displays users who have submitted at least one answer for this particular exercise. Once a user activity state is marked as *submitted* it will never revert to one of the previous activity states. This decision is based on the assumption that is more important for presenters to know that a user has submitted at least once than knowing their are currently in one of the other three states. Defaults to green.

To display the gradient of change for a certain activity state (e.g submitted), we use a slightly bigger progress bar of the same style in the background of the main progress bar. The faster the animation from left to right, the more activity occurred for a given state in the last 5 seconds. The presenter can choose if they want instant updates for the progress bar or if the progress bar should update simultaneously with the momentum effect by using the `directUpdate` property. At the end of the progress bar there are two numbers separated by a forward slash. The number in the left of the slash denotes the number of submissions, whereas the number on the right of the slash corresponds to the number of users that are connected to the live presentation. The number of connected users may be less than the number of submissions if users disconnect after submitting the associated exercise.

In the bottom of the example shown in Figure A.7, we can see that there are 5 people connected, 2 of which have submitted an answer to the exercise. There is also 1 viewer in the *working*, 1 viewer in the *focused* and 1 viewer in the *idle* state as indicated by the orange, light blue and gray colors respectively.

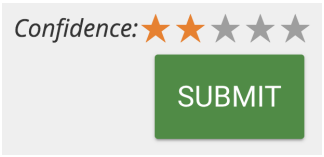
Listing A.3 shows the code that results in Figure A.7. Notice the `confidence` attribute.

```
1 <asq-exercise confidence>
2   <!-- One or more question types here -->
3 </asq-exercise>
```

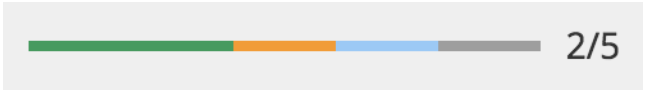
Listing A.3. Sample code for an `<asq-exercise>` element.

A.3.2 Welcome element

`<asq-welcome>` is an element to facilitate the connection of viewers in a presentation. It is usually placed on the first slide of a presentation and may be



Confidence rating and submit button in Audience View.



Exercise activity overview progress bar in presenter view.

Figure A.7. Example of an `<asq-exercise>`.

repeated on the corner of the presentation for the following slides. The presenter view renders the URL of the live presentation, a Quick Response (QR) code for easy access on camera equipped device and the number of connected users as shown in the bottom of Figure A.8. The viewer view displays the status of the connection to the `ASQ` server (top of Figure A.8).

Audience View

Presenter View

Figure A.8. Example of an `<asq-welcome>` element.

A.3.3 Teacher Utilities

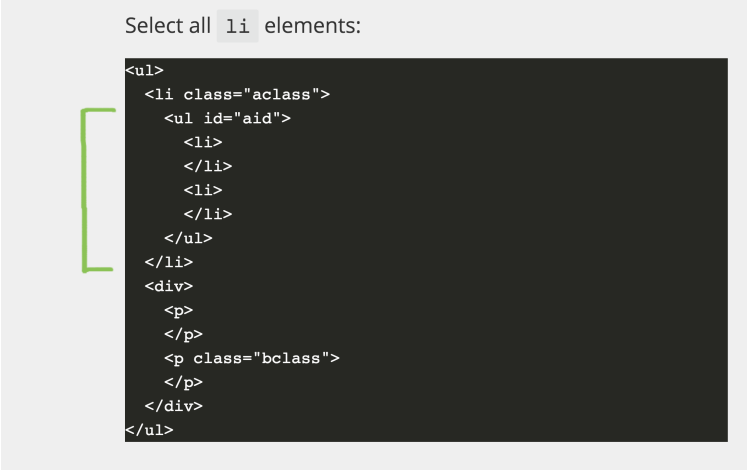
To annotate slides, presenters have the `<asq-canvas>` [237] widget at their disposal. Authors need to include this element at the top of their presentation markup as shown in the example of Listing A.4.

```
1 <body>
2    <asq-canvas></asq-canvas>
3    <!- rest of presentation markup goes here ->
4 </body>
```

Listing A.4. Example of how to include `<asq-canvas>` in a presentation

The presenter view features a toolbar to select the color and thickness of the canvas brush as shown at the bottom of Figure A.9. Presenters can erase parts of a drawing or delete it all together. They can also download a drawing in the Portable Network Graphics (PNG) format. Drawings are broadcast to the viewers in real time. Canvas content is scaled to fit their screen resolution and slideshow width/height ratio, as demonstrated at the top of Figure A.9.

Presenters can also create new empty slides which they can use as drawing space or a printout of the presentation slides using the same toolbar.



Audience View

Presenter View

Figure A.9. Example of an `<asq-canvas>` element.

A.3.4 Common elements

It is often the case that different question types need to implement similar functionality. For example, a lot of question types have a *stem* to provide the beginning ot the question. We have abstracted such functionality into elements to be used within the ASQ question types.

`<asq-stem>`

The stem of a question is a term mostly used for multiple choice questions. However, it applies to many other question types where some sort of introductory statement is made. It refers to the introductory part of an item's text that presents a problem to be solved, a question asked of the respondent, or an incomplete statement to be completed. Authors can choose to use the `<asq-stem>` element to specify the stem of a question in ASQ. This element is merely used for semantic purposes; by default it does not have any styling associated with it.

`<asq-solution>`

The `<asq-solution>` element is used to define the solution of question type. Any content enclosed by the `<asq-solution>` tags is considered to be the solution of the parent question element. `<asq-solution>` is agnostic about the content type of the solution as different question types have different solution content types such as plain text, numeric, JavaScript Object Notation (JSON) or JavaScript. This element is used when a presentation is uploaded to persist the solution in the database. Thereafter, it is removed from the markup of the presentation to avoid leaking the solution during the present or answer phase. When appropriate, it can be retrieved and stream to the presenters or viewers, for example during the feedback time.

A.4 Question Types

Each question in ASQ is of a certain `question type`. We have implemented 14 question types that are summarized in Table 3.1. We will present each one of them in dedicated subsections.

A.4.1 Text (<asq-text-input-q>)

The *text* question is a generic cued/free recall format question type where students are asked to come up with textual answer a question. It is implemented with the <asq-text-input-q> element that displays a simple single line text input question type. Authors can provide a solution string using the <asq-solution> element. Regular expressions as a solution are supported. Submissions are grouped lexicographically.

<asq-text-input-q-stats>

This auxiliary element can be used alongside an <asq-text-input-q> question to display the grouped submissions on the beamer. Authors need to set the for-id attribute to the id of the <asq-text-input-q> element they wish to display grouped results for.

Example and Usage

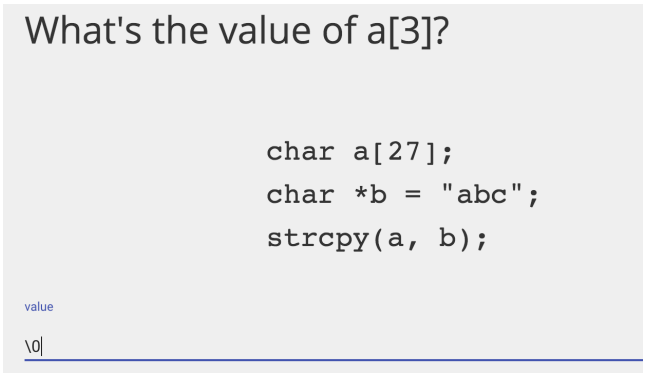
Listing A.5 shows the code which authors have to include in their slide to create the viewer and presenter versions of <asq-text-input-q> shown in Figure A.10. Respectively, Listing A.6 and Figure A.11 show the markup and rendered versions for a <asq-text-input-q-stats> element that will display grouped results for the aforementioned question.

```
1 <asq-text-input-q label="value" id="text-q">
2   <asq-stem><h3>What's the value of a[3]?</h3></asq-stem>
3   <pre><code>
4     char a[27];
5     char *b = "abc";
6     strcpy(a, b);</code></pre>
7   <asq-solution hidden>\0</asq-solution>
8 </asq-text-input-q>
```

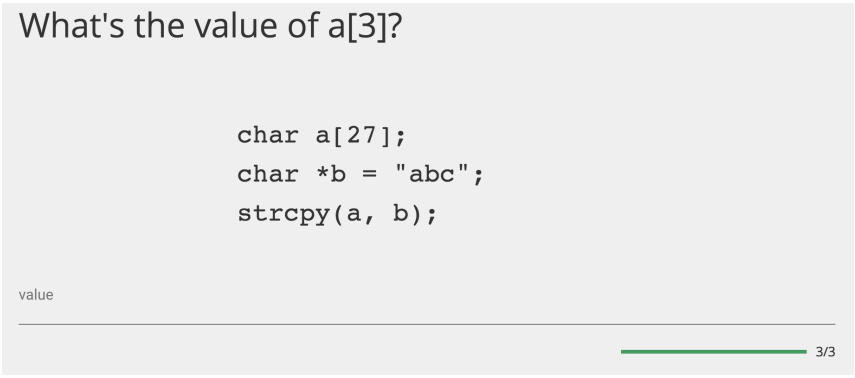
Listing A.5. Example of an <asq-text-input-q> question type element

```
1 <asq-text-input-q-stats for="text-q"></asq-text-input-q-stats>
```

Listing A.6. Example of an <asq-text-input-q> auxiliary element that will display grouped results for the element in Listing A.5

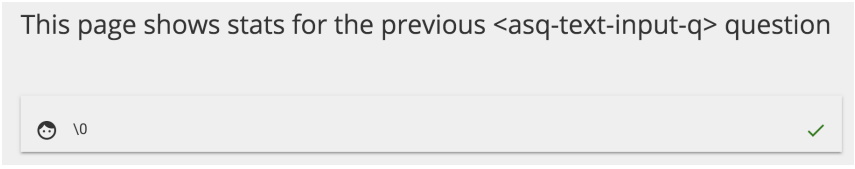


Audience view

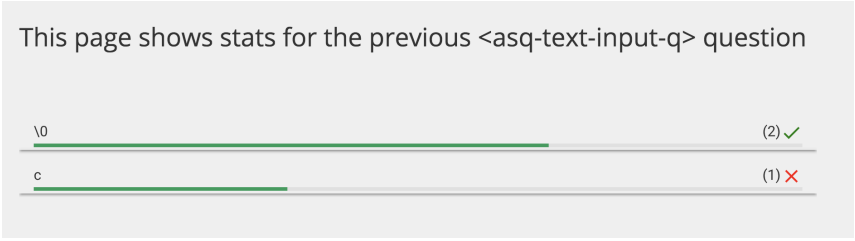


Presenter view.

Figure A.10. Example of an `<asq-text-input-q>` question type element.



Audience view



Presenter view

Figure A.11. Example of an auxiliary `<asq-text-input-q-stats>` element that displays statistics for the submissions of a `<asq-text-input-q>` element.

A.4.2 Multiple Choice (<asq-multi-choice-q>)

Recognition format questions are associated mostly with *multiple choice* questions. Although they do not achieve the deep learning associated with more effortful retrieval [114; 159] they are, in general, faster to answer and can still extract a shallow image of the level of comprehension of a subject. We created <asq-multi-choice-q> to support multiple choice questions. Options can be defined by using the <asq-option> element. Authors can choose to allow only one option to be selected (options render with a radio button next to them) or many (options render with checkboxes next them) by using the multi attribute.

In cases when the available list of options do not suffice, it is possible to make the question open-ended with an option to display a text field when checked, so that the viewers can provide their own custom answer. To configure an option for this functionality, the authors have to set the optional binary attribute is-other (Listing A.8).

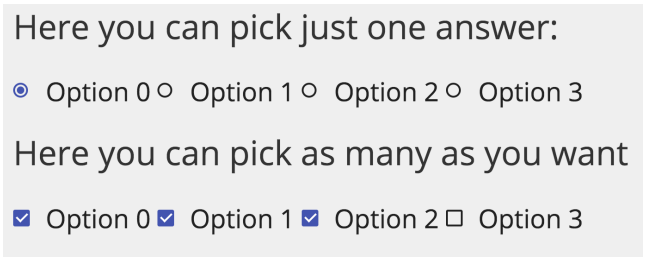
Student submissions can be visualized in the presenter view as horizontal or vertical histograms as shown in Figure A.12. If an option has the is-other attribute set, the pertinent text submissions are lexicographically grouped and a histogram for each group is drawn on the screen. By default, the histograms are hidden to prevent students from getting influenced by their classmates submissions. Toggling the "Show Solution" button on, will display or hide the histograms.

Listing A.7 shows the markup necessary to create two <asq-multi-choice-q> questions. The first allows only on option to be selected while the second allows any number of options. The rendered questions for viewers and presenters is presented in Figure A.12. Notice the the toggle button in the presenter view that displays/hides the submission histograms.

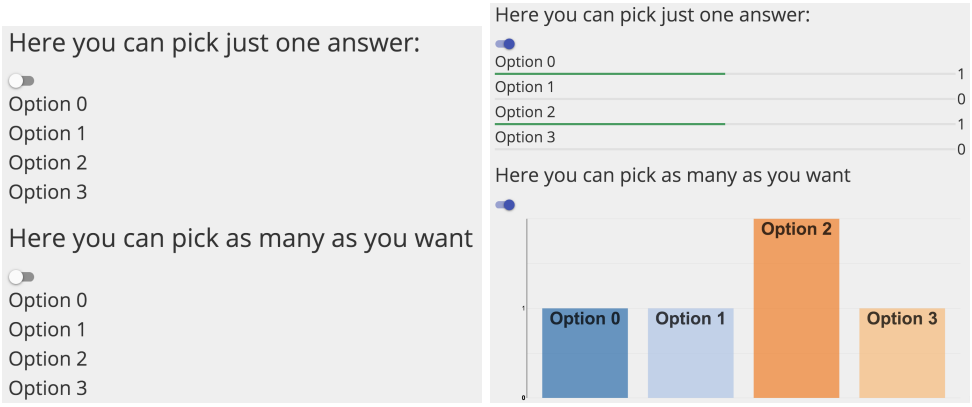
```
1  <asq-multi-choice-q selected="op3">
2    <asq-stem><h3>Here you can pick just one answer:</h3></asq-stem>
3    <asq-option name="op0">Option 0</asq-option>
4    <asq-option name="op1">Option 1</asq-option>
5    <asq-option name="op2">Option 2</asq-option>
6    <asq-option name="op3">Option 3</asq-option>
7  </asq-multi-choice-q>
8  <asq-multi-choice-q selected-values='["op1", "op2"]' multi
       stats-layout="vertical">
9    <asq-stem><h3>Here you can pick as many as you want</h3></asq-stem>
10   <asq-option name="op0">Option 0</asq-option>
11   <asq-option name="op1">Option 1</asq-option>
12   <asq-option name="op2">Option 2</asq-option>
13   <asq-option name="op3">Option 3</asq-option>
```

```
14 </asq-multi-choice-q>
```

Listing A.7. Example of two `<asq-multi-choice-q>` question types. The top one allows for just one option to be selected, while the bottom one allows for any number of options.



Audience view



Left: presenter view without displaying histograms. Right: presenter view with statistics being displayed.

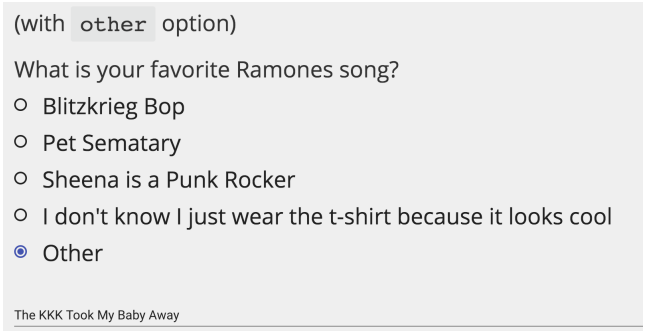
Figure A.12. Example of two `<asq-multi-choice-q>` question types.

Listing A.8 displays an `<asq-multi-choice-q>` where the last option has the attribute `is-other` set. Figure A.13 shows the rendered result. Notice the text box in the viewer view when the “Other” option is checked and the grouped “Other” submissions in the presenter view.

```
1 <asq-multi-choice-q>
2   <asq-stem>What is your favorite Ramones song?</asq-stem>
3   <asq-option name="a">Blitzkrieg Bop</asq-option>
4   <asq-option name="b">Pet Sematary</asq-option>
5   <asq-option name="cs">Sheena is a Punk Rocker</asq-option>
6   <asq-option name="d">I don't know I just wear the t-shirt because it looks
        cool</asq-option>
```

```
  <asq-option is-other name="other">Other</asq-option>
</asq-multi-choice-q>
```

Listing A.8. Example of two `<asq-multi-choice-q>` question types. The top one allows for just one option to be selected, while the bottom one allows for any number of options.



Audience view

Presenter view

Figure A.13. Example of a `<asq-multi-choice-q>` that contains an `<asq-option>` with the `is-other` attribute set.

A.4.3 Code (<asq-code-q>)

A lot of our teaching is focused in Programming Languages. When teaching concepts (for example for-loops or closures in JavaScript) we want our students to go through the process of creating code snippets to help them absorb the material better. To this end we create the *code* question type which essentially is an open text question (cued/free recall) with language-specific syntax highlighting. The <asq-code-q> question type displays a code editor with syntax highlighting as shown on the top of Figure A.14. It's based on the [ace18] library that supports highlighting according to the syntax of over 100 programming languages.

The presenter view displays a list with all viewer submissions on the right. Clicking on these submissions will load the text of that submission on the code editor of the presenter view. In our example, the first submission is selected and displayed on the code editor. Authors can customize the theme, mode and font size of the editor using the theme, mode and font-size properties accordingly. To provide some initial text to be displayed as the editor content, authors should add a <code> element with the desired textual content inside the <asq-code-q> element as demonstrated in Listing A.9. In this specific example, the <asq-code-q> element will display the text "for()".

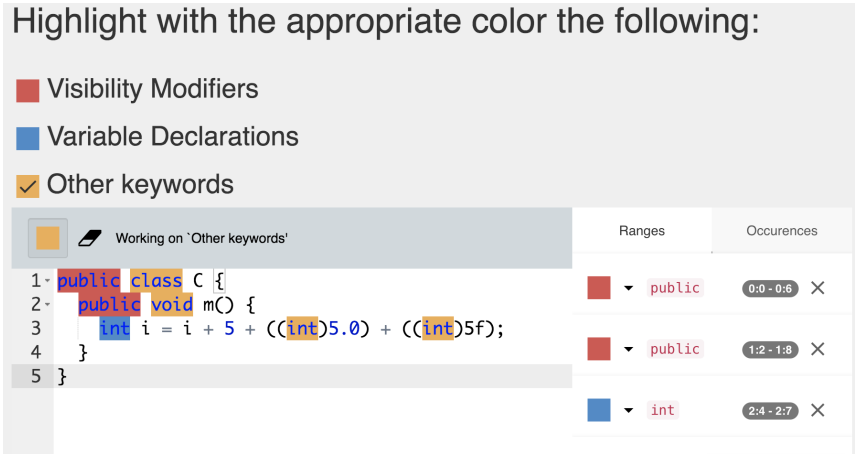
```
1 <asq-code-q theme="monokai" mode="javascript" font-size="0.8em">
2   <asq-stem><h3>Write a simple for loop in JavaScript </h3></asq-stem>
3 <code>for()</code>
4 </asq-code-q>
```

Listing A.9. Example of an <asq-code-q> question type

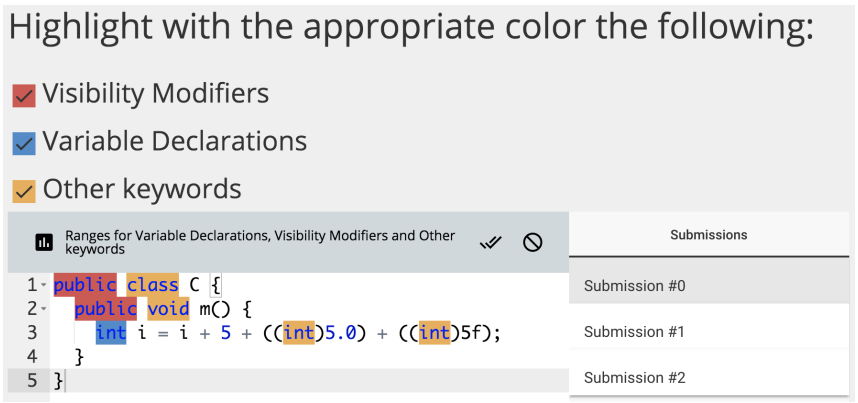
Figure A.14. Example of an `<asq-code-q>` question type. The presenter view is showing 3 submitted answers.

A.4.4 Highlight (<asq-highlight-q>)

The *highlight* question type is used to group under the same highlight color portions of text that belong to the same category. Examples include identifying categories of syntax constructs in a programming language (e.g., variable scoping or function declarations vs. calls) and spotting text with correct/wrong syntax. Our implementation is the <asq-highlight-q> element where learners use it to highlight portions of text with a specific color according to some given tasks. For example, the question in Figure A.15, specifies three tasks: the first one asks to highlight "Visibility Modifiers" with a red color; the second one "Variable Declarations" with a red color; and the third one to "Other keywords" with a yellow color. Viewers select the appropriate colors and highlight the text accordingly. <asq-highlight-q> is based on Ace editor.



Audience view



Presenter view

Figure A.15. Example of an <asq-highlight-q> question type.

Authors can configure the font size, syntax mode and theme using the `font-size`, `mode` and `theme` properties respectively. They can also provide a solution using the `<asq-solution>` element. A solution consists of an array of highlight ranges for each colored task encoded in JSON format. Listing A.10 shows the code that was used to create the question of Figure A.15.

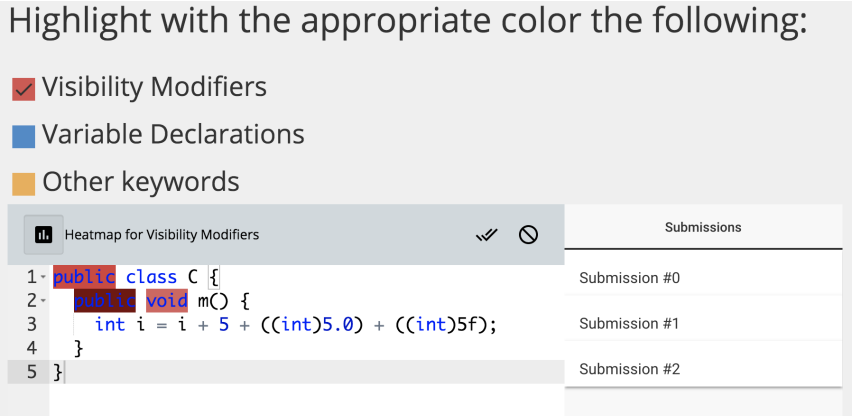
```
<asq-highlight-q theme="textmate" mode="java" font-size="0.65em">
  <asq-solution>{"d9534f":[{"start":{"row":1,"column":2},"end":{"row":1,
      "column":8},"id":216},{"start":{"row":0,"column":0},"end":{"row":0,
      "column":6},"id":222}],"428bca":[{"start":{"row":2,"column":4},
      "end":{"row":2,"column":7},"id":229}],"f0ad4e":[{"start":{"row":0,
      "column":7},"end":{"row":0,"column":12},"id":236},{"start":{"row":1,
      "column":9},"end":{"row":1,"column":13},"id":243},{"start":{"row":2,
      "column":22},"end":{"row":2,"column":25},"id":250},{"start":{"row":2,
      "column":35},"end":{"row":2,"column":38},"id":256}]}</asq-solution>
    <asq-stem><h3>Highlight with the appropriate color the
        following:</h3></asq-stem>
    <asq-hl-color-task color="d9534f">Visibility Modifiers</asq-hl-color-task>
    <asq-hl-color-task color="428bca">Variable Declarations</asq-hl-color-task>
    <asq-hl-color-task color="f0ad4e">Other keywords</asq-hl-color-task>
    <code>public class C {
  public void m() {
    int i = i + 5 + ((int)5.0) + ((int)5f);
  }
}</code>
</asq-highlight-q>
```

Listing A.10. Example of an `<asq-highlight-q>` question type

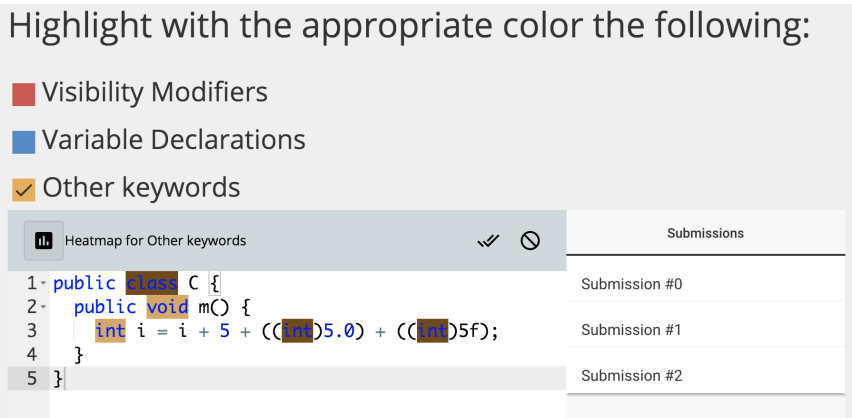
Presenters have three ways at their disposal to provide feedback to students. The first one, similar to the one in `<asq-code-q>`, displays a submission list. Presenters can select and display the submission of a student by clicking an entry in the list as shown on the bottom of Figure A.15. The second way is to display a heatmap designed to give presenters the "big picture" of submission per highlighting task. The more a character of text has been highlighted from the audience in a specific color (for the pertinent task), the darker the shade of the color this character is highlighted with in the heatmap view. Figure A.16 shows the heatmaps after three viewer submissions for the red task on the top and yellow task on the bottom. The last way is to display the reference solution.

Highlight Editor

To avoid writing repetitive markup and JSON, authors can use an GUI editor to create `<asq-highlight-q>` questions. Using the `<asq-highlight-q>` editor



Heatmap for the task in red color



Heatmap for the task in yellow color

Figure A.16. Heatmaps in `<asq-highlight-q>` presenter view.

(Figure A.17), authors can provide a stem, type the text to highlight, and create the highlighting tasks. They can configure the font size, code editor theme and the syntax highlighting theme. To create a solution, authors have to highlight the text ranges of their choice. Once the question is ready, the editor provides the HTML markup necessary to create an `<asq-exercise>` containing the newly created question, as shown in Figure A.18. Authors may copy this markup inside a slide of their presentation. The editor can also output just the solution.

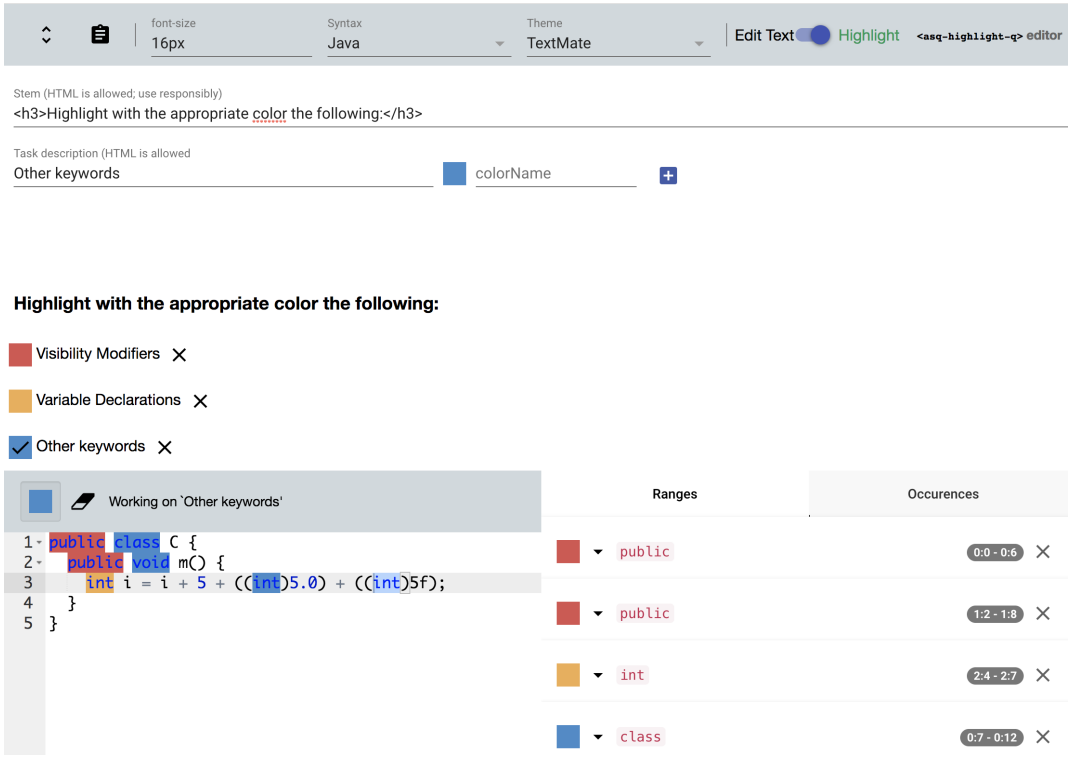


Figure A.17. The `<asq-highlight-q>` editor GUI. Here shown in progress of creating the question of Listing A.10

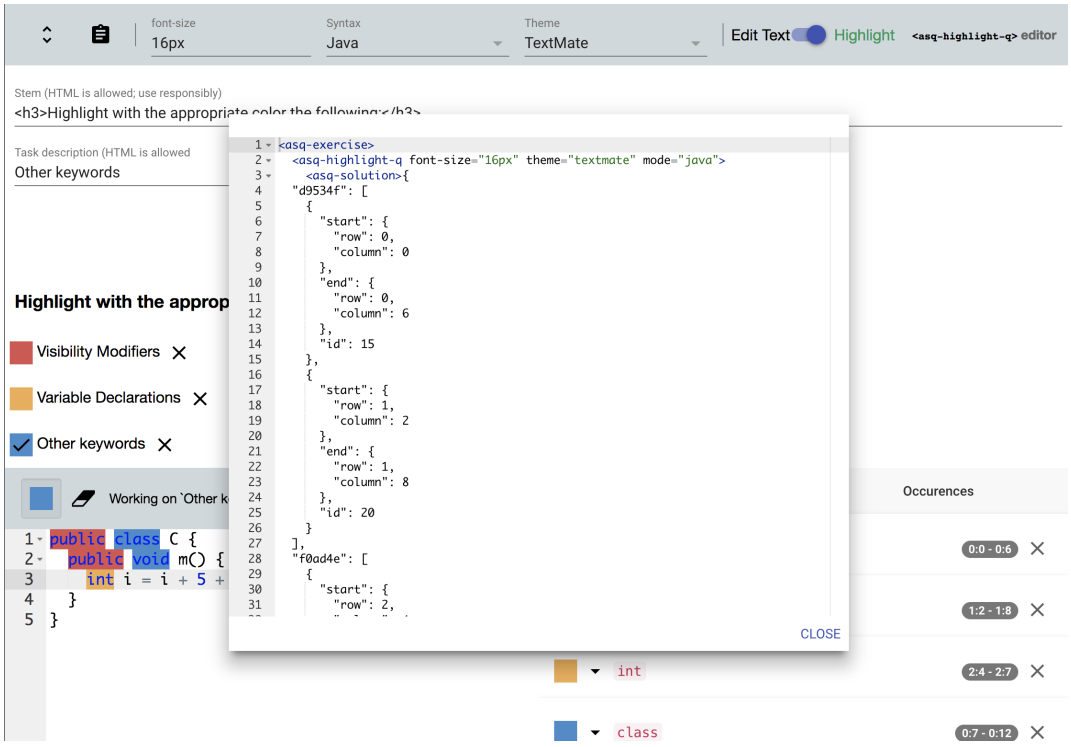


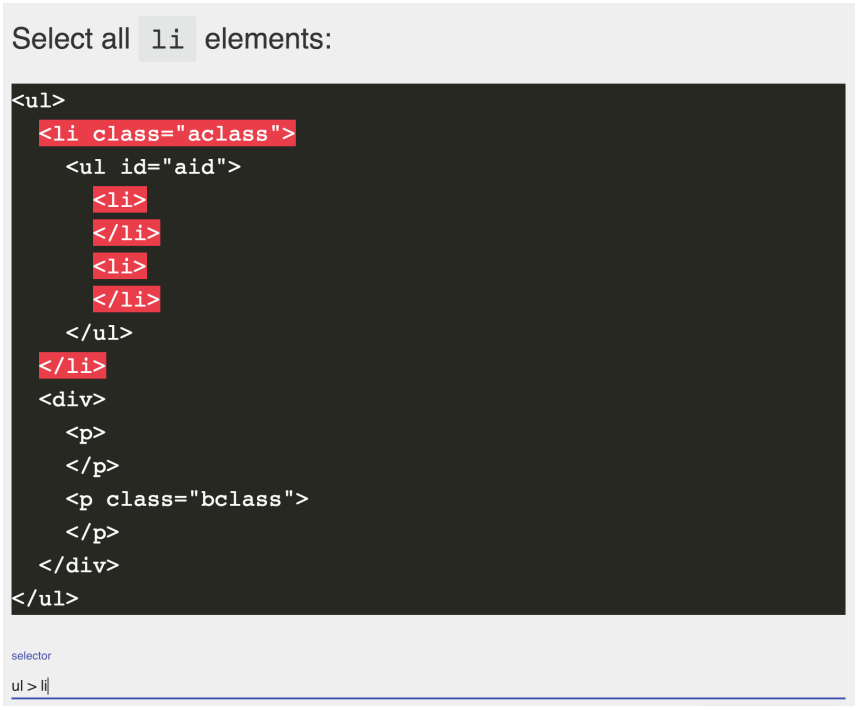
Figure A.18. HTML markup output for an `<asq-highlight-q>` question that includes the solution

A.4.5 CSS Select (`<asq-css-select-q>`)

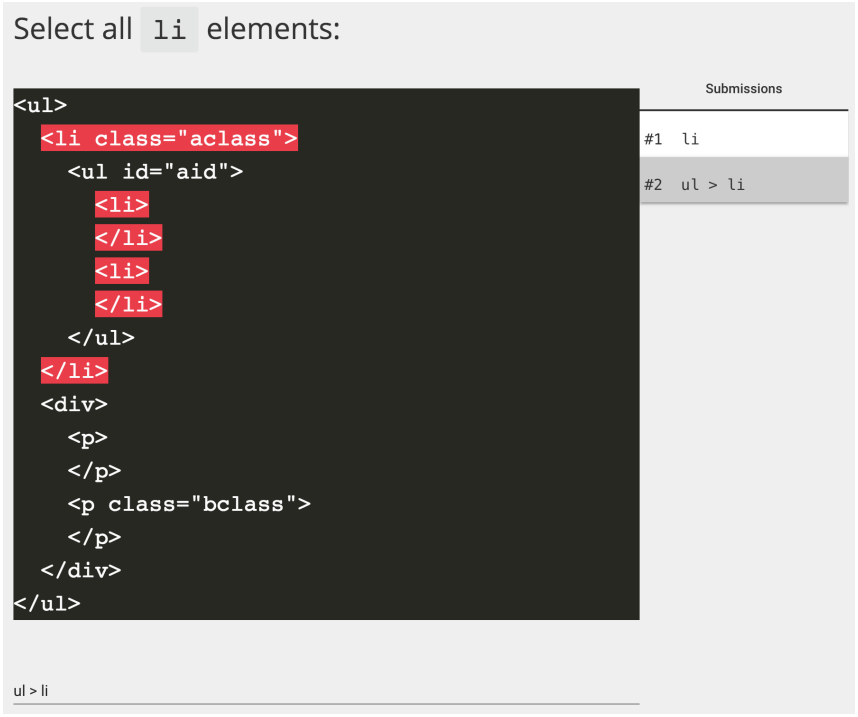
In our courses where we teach Web Technology concepts, a big sticking point for learners is how CSS Selectors work [mdn18a], to paraphrase the popular proverb, "an interactive example is worth a 1000 pictures" and we find that a *live programming* [25] demonstration that highlights the fragments of an HTML document that match a specific selector can help a great deal in understanding this concept. Moreover, due to the nature of CSS selectors, there is more than one correct solution. Comparing different solutions helps the students realize the alternatives and lead to discussions on best practices. *CSS select* is one of our first live programming question types (cued/free recall) (`<asq-css-select-q>` element) to teach students and works as follows: authors provide some HTML markup using the `htmlcode` attribute as shown in the example of Listing A.11, that will be used to practice selectors on. From this markup an HTML node tree will be created automatically that displays only one tag per level as in Figure A.19. Viewers can type CSS selectors in the provided text input. Each time they type, the selector expression gets evaluated and the matching nodes of the HTML tree are highlighted in real time in fuchsia color as shown in Figure A.19. This way students get immediate feedback of what part of the HTML tree their selectors match. Similar to the `asq-code-q`, the presenter view displays a list of all submissions. When the presenter clicks on a submission, the corresponding submission will be applied on the question type and will be displayed in the beamer.

```
1 <asq-css-select-q htmlcode="<ul><li class='aclass'><ul
      id='aid'><li></li><li></li></ul></li><div><p></p><p
      class='bclass'></p></div></li></ul>">
2   <asq-stem><h3>Select all <code>li</code> elements:</h3></asq-stem>
3 </asq-css-select-q>
```

Listing A.11. Example of an `<asq-css-select-q>` question type



Audience view



Presenter view

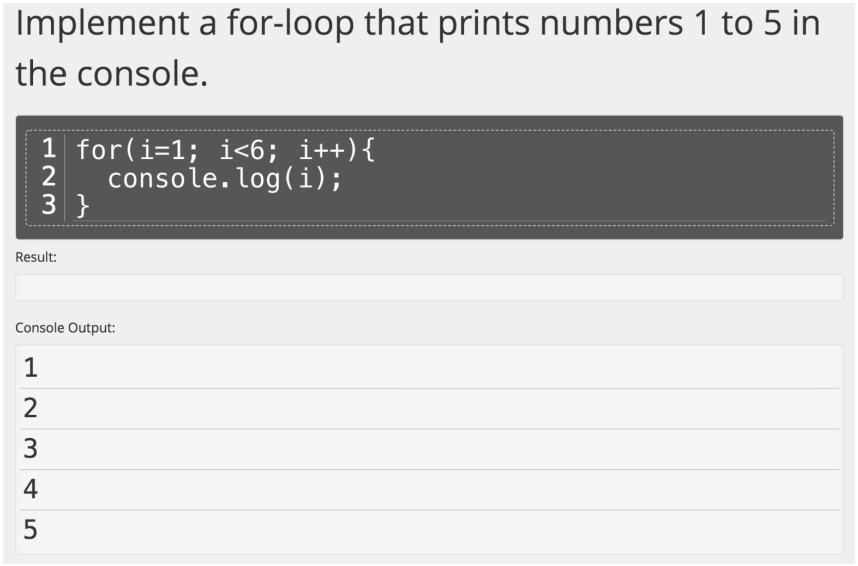
Figure A.19. Example of an `<asq-css-select-q>` question type.

A.4.6 Live JS(<asq-js-eval-q>)

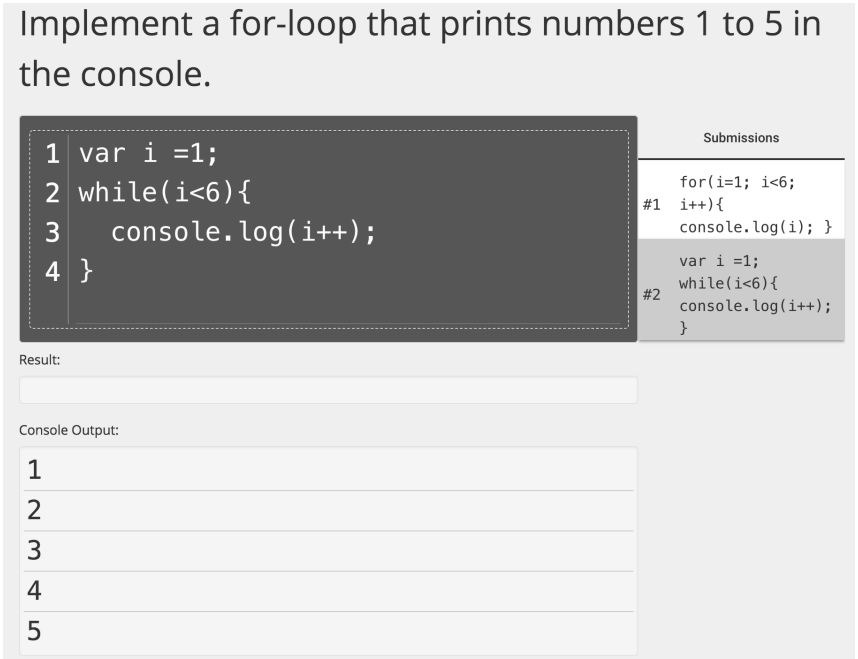
While the code question type facilitates typing code highligting text according to the syntax of a programming language, it does not execute the submitted code. The browser allows to evaluate JavaScript so we took advantage of this features to give instantaneous feedback to our students on their program execution and offload part of the assessment task that normally burdens the teacher to them. *Live JS* (cued/free recall), is implemented with the <asq-js-eval-q> element. It is the second question type after <asq-css-select-q> that supports immediate evaluation of the code (live programming). Viewers get a simple text editor without syntax highlighting to practice their JavaScript skills, like shown in the top of Figure A.20. What they type gets evaluated (with a delay of 500 milliseconds after the last character was typed) continuously and the return values, errors and console logs of the execution are reported to the viewer. The evaluation occurs in a separate browser thread using a Web Worker [WW18] to prevent performance degradation and crashes of the main JavaScript thread that runs in the browser window. The worker will stop the code execution if the code runs for too long. Presenters have access to the submission list. They can select, display on the code editor and edit submissions to provide feedback to their viewers.

```
1 <asq-js-eval-q>
2   <asq-stem><h3>Implement a for-loop that prints numbers 1 to 5 in the
        console.</h3></asq-stem>
3 </asq-js-eval-q>
```

Listing A.12. Example of an <asq-js-eval-q> question type



Audience view



Presenter view

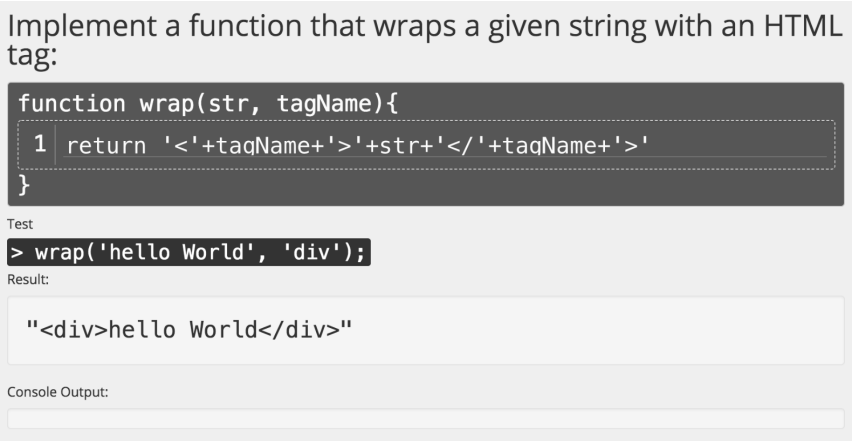
Figure A.20. Example of an `<asq-js-eval-q>` question type.

A.4.7 JS Function<asq-js-function-body-q>

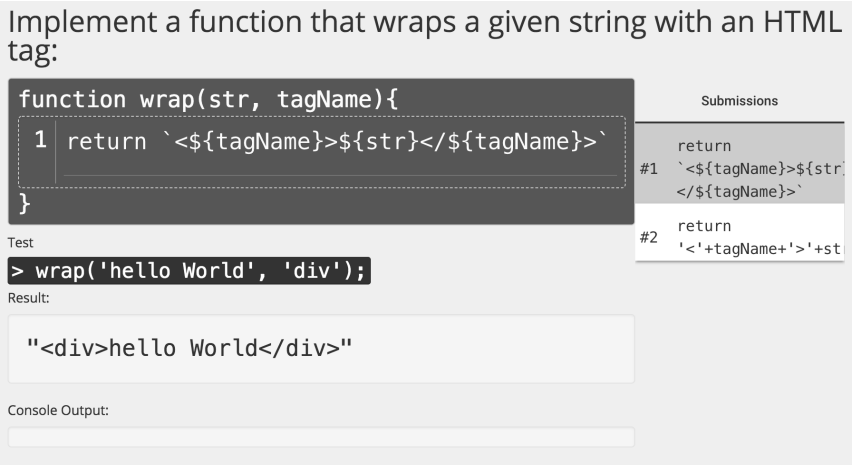
JS function(<asq-js-function-body-q> element) implements a specialization of Live JS that focuses in learning and verifying the output of JavaScript functions. Authors can specify the name and arguments of a function and a test expression by using the function-name and test-exp attributes respectively, as shown in the example of Listing A.13. Students have to implement the body of the function. As they type their code, the test expression is invoked and the return value of the function, the console statements and errors are displayed on the screen similar to <asq-js-eval-q>. Figure A.21 shows the rendered question that results from the source code of Listing A.13. The presenter view features the same functionality as <asq-js-eval-q>.

```
1 <asq-js-function-body-q function-name="wrap(str, tagName)"
      test-exp="wrap('hello World', 'div');">
2   <asq-stem><h3>Implement a function that wraps a given string with an HTML
        tag:</h3></asq-stem>
3 </asq-js-function-body-q>
```

Listing A.13. Example of an <asq-js-function-body-q> question type



Audience view



Presenter view

Figure A.21. Example of an `<asq-js-function-body-q>` question type.

A.4.8 SQLite (<asq-sqlite-q>)

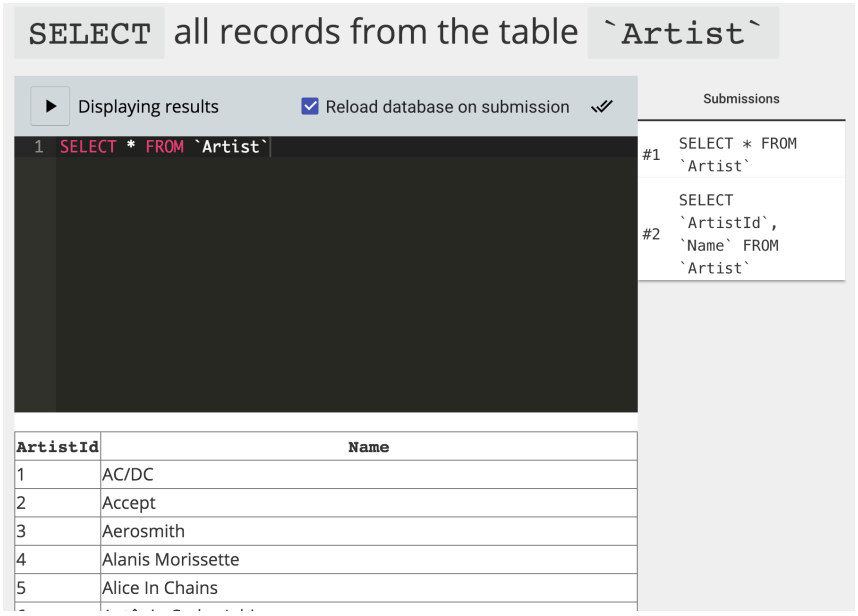
For the deployment of ASQ in a course teaching Web technology and database concepts (see section 5.1.2) we needed a way for students to run SQL [51] during lectures in large classroom with more that 200 participants. Instead of relying on setting up external servers to support this functionality which would require to spend time and economical resources for a system with not optimal response times for query execution we opted to execute the queries inside the browser based on recent developments in compiling C/C++ projects to efficient JavaScript (using the Emscripten compliler [191]). *SQLite* (<asq-sqlite-q> element) is the fourth question type (cued/free recall) that supports instant code evaluation within the browser. It is designed to allow the execution of SQLite [132] queries in the browser without the need for an external server. It takes advantage of the *sql.js* [Zak16] library, a project that compiles SQLite to JavaScript through Emscripten. Conceptually similar to <asq-js-eval-q>, SQLite runs in a Web Worker thread to prevent the browser interface from being unresponsive or freezing completely. Since there is an SQLite instance running in the browser of the viewer, there is no need for server-side sandboxed SQLlite infrastructure which increases complexity and adds security and availability concerns. To supply a seed SQLite database, authors can set the db-url attribute to an appropriate URL. A reference solution may be provided in the form of a SQLite query expression enclosed inside an <asq-solution> element. This is demonstrated in the example question of Listing A.14.

Both viewers and presenters can type queries in the supplied Ace editor and evaluate them by clicking the top left button with the "play" triangle symbol. Query results are displayed below the editor. Presenters may choose and display a viewer submission from the list of submissions on the right and edit it to provide feedback. The can also display the reference solution by clicking on the dedicated button. Figure A.22 shows the presenter and viewer views for the source code of Listing A.14.

```
1 <asq-sqlite-q db-url="assets/Chinook_Sqlite.sqlite" font-size="0.6em">
2   <asq-stem><h3><code>SELECT</code> all records from the <code>TABLE</code>
        <code>‘Artist‘</code></h3></asq-stem>
3   <code></code>
4   <asq-solution>SELECT * FROM ‘Artist‘</asq-solution>
5 </asq-sqlite-q>
```

Listing A.14. Example of an <asq-sqlite-q> question type

Audience view



Presenter view

Figure A.22. Example of an `<asq-sqlite-q>` question type.

A.4.9 Rate (<asq-rating-q>)

Rate, implemented with the <asq-rating-q> element, is a simple rating question type that displays items to be rated. It is meant as fast and easy way to provide rating on a subject. It can be used for both expressing opinion or as a recognition question. Currently only a star rating scale of 0 to 5 stars (with half points) is supported. This needs to specified in the type attribute as in the example of Listing A.15. Each item's markup must be enclosed in an <asq-rating-item>.

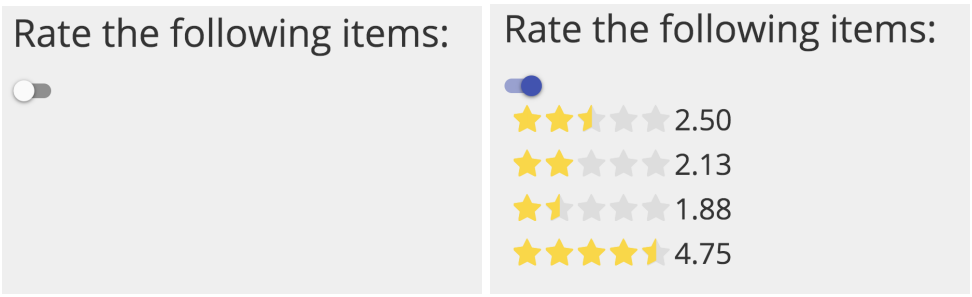
Viewers can rate each item by clicking on the star that corresponds to their rating (see top of Figure A.23). Clicking a star will select all stars on the left of the selected star and will sum 1 point for each star. Then, depending on where the click on the star happened one of the following will happen: clicking on the left half of a star will select half of the star and add 0.5 to the rating; clicking on the right half of the star will select the full start and add 1 point.

The presenter view can display the average ratings per item for all submissions by toggling the button as in the bottom right of Figure A.23. By default they are hidden to avoid biasing the audience as shown in the bottom left of Figure A.23.

```
1 <asq-rating-q type="stars">
2   <asq-stem><h3>Rate the following items:</h3></asq-stem>
3   <asq-rating-item name="item-1">Item to rate #1</asq-rating-item>
4   <asq-rating-item name="item-2">Item to rate #2</asq-rating-item>
5   <asq-rating-item name="item-3">Item to rate #3</asq-rating-item>
6   <asq-rating-item name="item-4">Item to rate #4</asq-rating-item>
7 </asq-rating-q>
```

Listing A.15. Example of an <asq-rating-q> question type

Audience view



Left: presenter view without displaying the average rating. Right: presenter view with statistics being displayed.

Figure A.23. Example of two `<asq-rating-q>` question types.

A.4.10 Classify (<asq-buckets-q>)

Often in our courses we teach concepts which can be classified into one or more classes. For example, during an HTTP 1.1 request there are headers [mdn18d] that are sent by the client (*Request*), headers that are sent by the server (*Response*) and headers that are sent by both parties (*General* or *Entity*). To provide such functionality we implemented *Classify* (<asq-buckets-q> element). It is a recognition format question type, created to help students express their understanding about concepts being related as part of the same category by grouping them together within the same bucket. It allows viewers place labels (items) into buckets (classes) allowing them to classify them. Currently each label can go into one bucket only.

In the simplest case – demonstrated in the source code of Listing A.16 – the authors should provide: 1. The attributes x-matchable and y-matchable. They accept a CSS selector that selects the *parent* HTML element of the *buckets* and *labels* respectively; and 2. a set of labels, a set of buckets that have the name attribute set to a distinct value and conform to the CSS selectors described before.

```
1  <asq-buckets-q x-matchable="div[buckets]" y-matchable="div[labels]">
2
3    <asq-stem><h3>Drag the labels (cities names) to matching
         buckets.</h3></asq-stem>
4
5    <div buckets>
6      <div name="Europe" class="bucket">Europe</div>
7      <div name="North America" class="bucket">North America</div>
8      <div name="Middle East" class="bucket">Middle East</div>
9    </div>
10
11   <div labels>
12     <div name="New York" class="label">New York</div>
13     <div name="London" class="label">London</div>
14     <div name="Zurich" class="label">Zurich</div>
15     <div name="Jerusalem" class="label">Jerusalem</div>
16     <div name="Chicago" class="label">Chicago</div>
17     <div name="Lugano" class="label">Lugano</div>
18   </div>
19 </asq-buckets-q>
```

Listing A.16. Example of an <asq-buckets-q> question type

In this example, the buckets' parent is a element with a buckets attribute. Likewise, the labels' parent is a element with a labels attribute. This setup will allow up to one label to go into a bucket as shown in the rendered

question of Figure A.24. If the user drags a label in a bucket that already has one, the existing label will be replaced with the new label.

Figure A.24. Example of the audience view for the `<asq-buckets-q>` question type.

The question type allows authors to customize a number of labels that can go into a bucket using the `mode` attribute: its value can be `1-1`, `1-m` or `1-1` where `m` denotes unlimited number of elements and `1` can be any positive integer. The attribute `attr-for-matched` denotes the name of the attribute of the labels and buckets that the matching will take place with. `matched-attribute` is the binary attribute that will be set on labels when they have been matched with a bucket. The `matched-class` attribute is a string that will be added to the `class` attribute of a matched element, to help style matched elements. Listing A.17 shows the source code for a more advanced example of a `<asq-buckets-q>` question, where we have configured all the available options. The resulting rendered question is displayed in Figure A.25. Notice that the buckets can accept up to three labels.

```
1  <asq-buckets-q
2    mode="1-3"
3    x-matchable="div[buckets]"
4    y-matchable="div[labels]"
5    attr-for-matched="s-name"
6    matched-attribute="matched"
7    matched-class="label-success">
8
9    <asq-stem><h3>Drag the labels (cities names) to matching
         buckets.</h3></asq-stem>
10   <div buckets>
11     <div s-name="Europe" class="bucket"><b>Europe</b></div>
12     <div s-name="North America" class="bucket"><b>North America</b></div>
13     <div s-name="Middle East" class="bucket"><b>Middle East</b></div>
14   </div>
```

```
15
16   <div labels>
17     <div s-name="New York" class="label label-primary">New York</div>
18     <div s-name="London" class="label label-primary">London</div>
19     <div s-name="Zurich" class="label label-primary">Zurich</div>
20     <div s-name="Jerusalem" class="label label-primary">Jerusalem</div>
21     <div s-name="Chicago" class="label label-primary">Chicago</div>
22     <div s-name="Lugano" class="label label-primary">Lugano</div>
23   </div>
24 </asq-buckets-q>
```

Listing A.17. Example of an `<asq-buckets-q>` question type

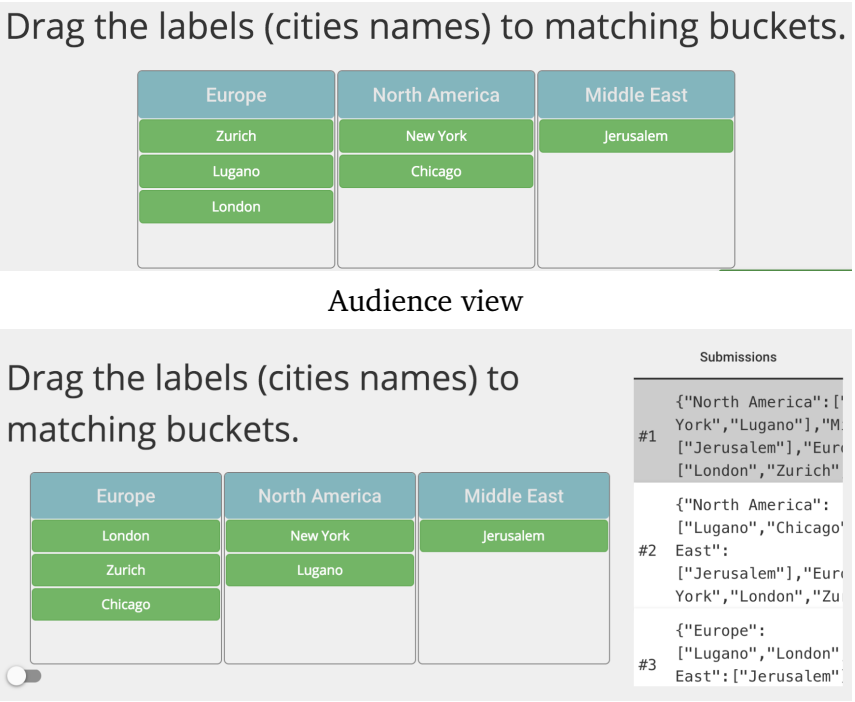


Figure A.25. More advanced example of an `<asq-buckets-q>` question type element.

Presenters can see a list of submissions and select it for display in a way similar to previous question types. They can also display how many times a label was matched to a bucket as shown in Figure A.26. As was the case with similar functionality in other question types, there is a toggle button to show the statistics to avoid biasing the audience towards specific answers.

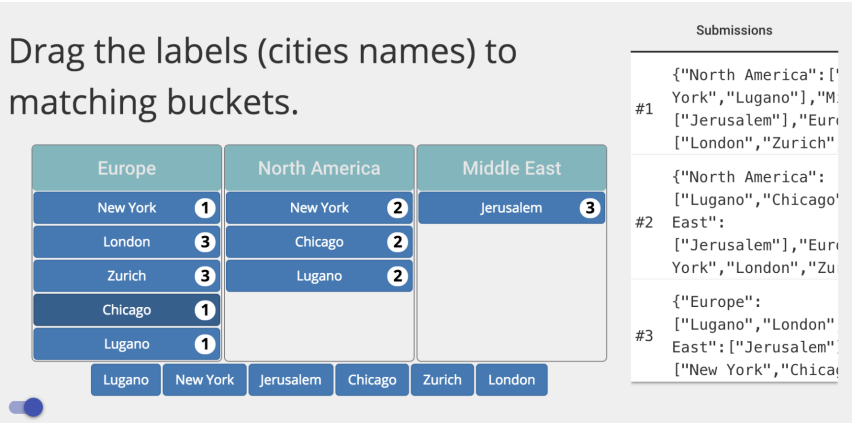


Figure A.26. Aggregated statistics displayed in the Presenter View for the `<asq-buckets-q>` question of Figure A.24

A.4.11 Order (<asq-order-q>)

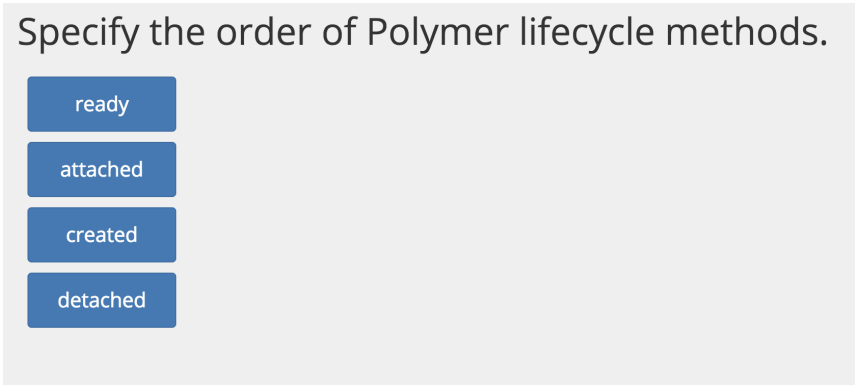
Order is a simple recognition format question type (implemented via the <asq-order-q> element to use when there is a need to order some items. Items may be reordered by dragging them and dropping them in the intended place.

The items that need to be ordered can be identified using the sortable attribute and supplying a CSS selector as its value. This is shown in the source code of Listing A.18. All elements that have the label attribute will be available for reordering.

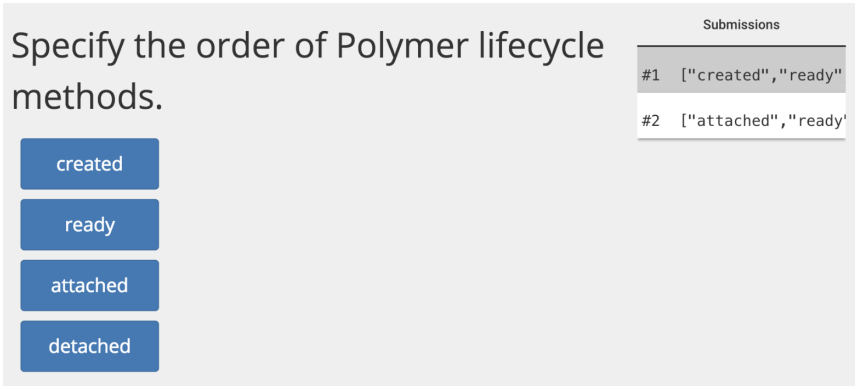
```
1 <asq-order-q sortable="[label]">
2   <asq-stem><h3>Specify the order of Polymer lifecycle methods.</h3></asq-stem>
3   <div label class="static" name="ready">ready</div>
4   <div label class="static" name="attached">attached</div>
5   <div label class="static" name="created">created</div>
6   <div label class="static" name="detached">detached</div>
7 </asq-order-q>
```

Listing A.18. Example of an <asq-order-q> question type element

Figure A.27 shows how the source code of Listing A.18 renders for the viewer an presenter views. As is the case with the majority of the question types, the presenter view displays a list of all submissions. Presenters can select and display an item from this list for classroom discussion.



Audience view



Presenter view

Figure A.27. Example of an `<asq-order-q>` question type element.

A.4.12 HTML Fiddle<asq-fiddle-q>

As instructors of Web Technologies we believe that "putting it all together" is a great way for students to demonstrate their deep understanding of the concepts we teach them. In terms of front-end Web page design, this translated to creating fully functional Web pages. A stepping stone is the creation of functionality that runs in a single page as it offers a first glance to how the pertinent technologies (HTML5, CSS3 and JavaScript) work in tandem. The *HTML fiddle* question type (cued/free recall) allows viewers to create a Web page and view the result immediately in the same browser window, similar to JSFiddle [ZK18]. It is the fith question that supports live programming. It features 4 resizable and collapsible panes for editing HTML, CSS and JavaScript and viewing the result, as shown in Figure A.28. The editing panes are based on the ACE editor library, while the result pane is a sandboxed `<iframe>` element. The HTML, JavaScript and CSS code are injected in the `<iframe>` page every time the viewer types in one of the editing panes.

Listing A.19 shows the source code for an example `<asq-fiddle-q>` question. To expand specific panes, authors can supply a JSON stringified array of the names of the panes as the value of the attribute `selected-panes`. In our example the HTML, JavaScript and Results panes will be open while the CSS panel will be collapsed. They may also configure the font size of the editors using the attribute `editor-font-size`.

```
1 <asq-fiddle-q selected-panes='["html", "js", "result"]'
      editor-font-size="1.2em;">
2   <asq-stem>
3     <h4>Create a header and print a message in the console when clicked</h4>
4   <asq-stem>
5 </asq-fiddle-q>
```

Listing A.19. Example of an `<asq-fiddle-q>` question type element

The list of submissions on the presenter view is supported in this elements as well.

For every supported editor authors can provide a seed value (code) to used as a starting point. These values can be provided inside a parameterized `<pre>` element that is a descendant of a `<template>` element. Te exhibit this with a code sample, Listing A.20 supplies initial values in all the editors. Notice that each `<pre>` element has an attribute which is the name of the target element: `html` for the HTML editor, `css` for the CSS edirot and `js` for the JavaScript editor. Figure A.29 shows the rendered result for viewers without any user input.

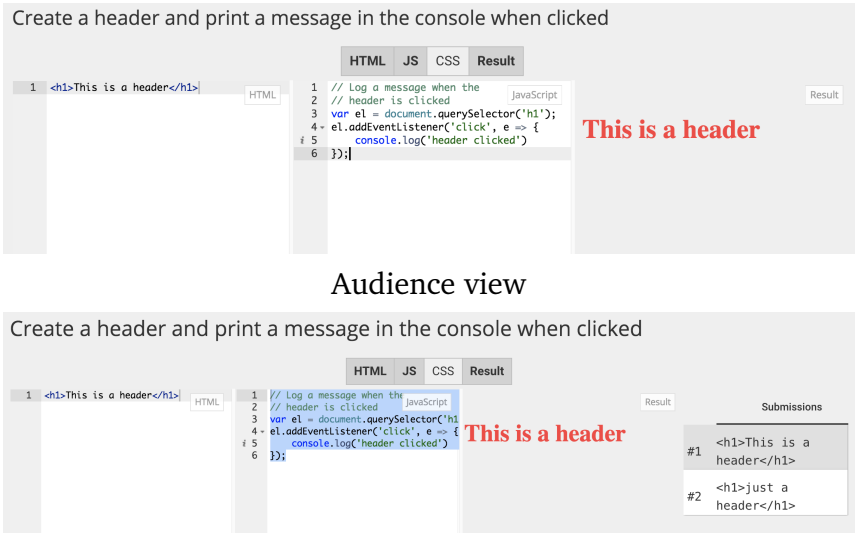


Figure A.28. Example of an `<asq-fiddle-q>` question type element.

```
<asq-fiddle-q selected-panes='["html", "js", "result"]'
    editor-font-size="1.2em;">
  <asq-stem><h4>Paint a circle in the canvas</h4></asq-stem>
  <template seed>
    <pre html><canvas id="c"></canvas></pre>
    <pre css>#c {width: 200px; height: 200px}</pre>
    <pre js>//Paint a circle in the canvas
var c = document.getElementById("c");</pre>
  </template>
</asq-fiddle-q>
```

Listing A.20. Example of an `<asq-fiddle-q>` question type element

Figure A.29. Audience view example of seeding the editors of an `<asq-fiddle-q>` question type element.

A.4.13 Java (<asq-java-q>)

ASQ can support non Web-native programming languages as well. *Java*, whose frontend is implemented via the <asq-java-q> elements is such a question type, that enables learners to create and execute Java [12] programs in a remote sandbox and receive the results in their browser [236]. This question type demonstrates the strengths of the plugin system of ASQ (see section 4.2 for more details) that allows asynchronous code execution on the backend. Authors can supply a suite of JUnit [BGSC18] tests to test submissions against. The errors or the output from successful execution are reported back to the client as shown in the audience view and the presenter view of Figure A.30.

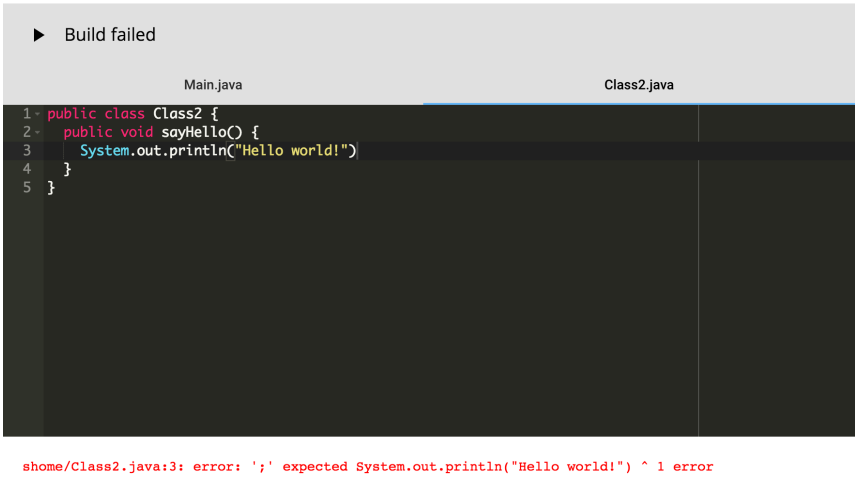
In this question type we support multiple files organized into java exercises (not to be confused with the <asq-exercise> element). Authors need to provide a JSON file (named files.json) that lists all the files of the exercise. The example in Listing A.21 displays the contents of the files.json) file for an exercise that contains three files: files.json, Main.java and Test.java. Listing A.22 shows the markup necessary to create a question with a Java exercise called "exercise1" (specified through the exercise-name attribute). The files of the exercise should be located inside a directory that has the same name as the exercise. This directory in turn should under a files directory inside the main directory of the presentation (Figure A.31).

```
1 {
2    "main": "Main.java",
3    "files": [
4       "Main.java",
5       "Test.java"
6    ]
7 }
```

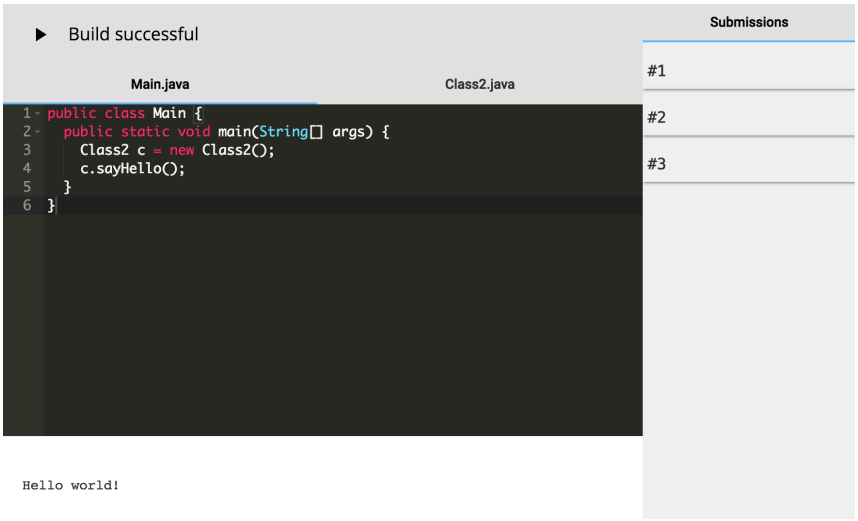
Listing A.21. Example of a files.json file that is used to specify the files of a Java exercise for the <asq-java-q> question type.

```
1 <asq-java-q exercise-name="exercise1">
2   <asq-stem>Exercise 1</asq-stem>
3 </asq-java-q>
```

Listing A.22. Example of an <asq-java-q> question type element



Audience view (displaying an compilation error)



Presenter view (displaying program backend)

Figure A.30. Example of an `<asq-fiddle-q>` question type element.

`<asq-java-q>` backend

To provide scalabity and security guarantees, tests for java submissions for `<asq-java-q>` are executed in a remote backend which is separate from the `ASQ` server. Each submission submitted from to `ASQ` is aggregated in the backend components of `<asq-java-q>` and pushed into a queue at the remote backend. Every time an item is popped from the queue, a new Docker [123] container is instantiated. The source code of the submission together with the author-specified JUnit tests

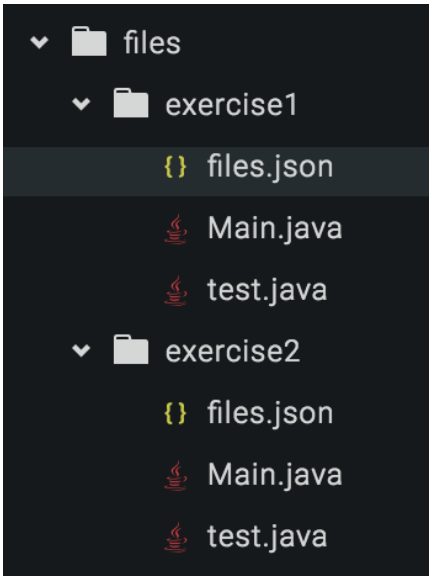


Figure A.31. Directory structure two support two Java exercises for the `<asq-java-q>` question type named ‘exercise1’ and ‘exercise2’.

are passed in to the container. The tests are executed and the results are sent back to `ASQ` via the question type plugin backend.

A.4.14 Diagram (<asq-diagram-q>)

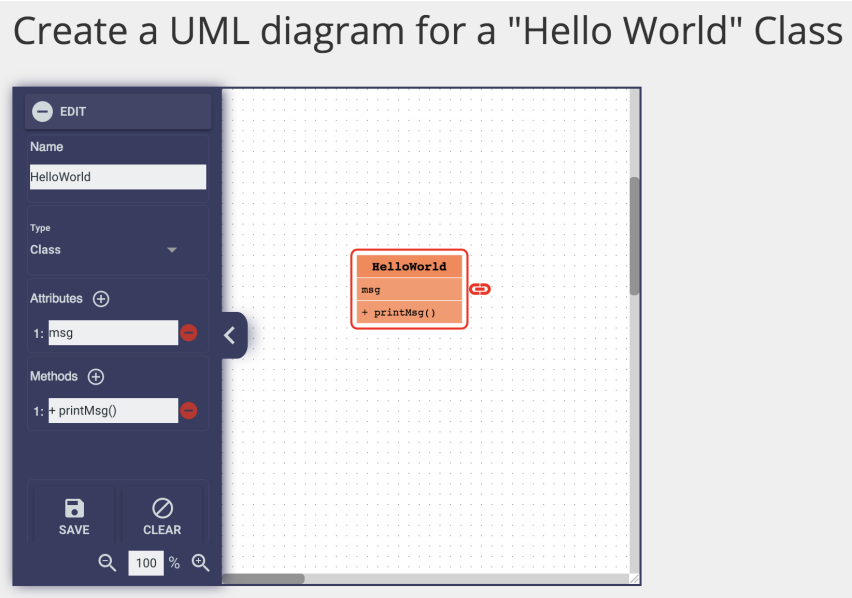
Diagrams play a prominent role in Computer Science and there are various types that are used in various subfields. In our courses we teach topics that cover Web technology; databases; software architecture and design; and business process modeling. We frequently found ourselves in the need for a question type where students could draw diagrams in pertinent formats. For example in a database course students need to learn how to abstract the data model using Entity–relationship (ER) [44] diagrams while in a business process modeling course, students should be able to model a business process using the Business Process Model and Notation (BPMN) [8]. *Diagram*, which is implemented via the <asq-diagram-q> element [239], is a question type that currently allows the creation of Unified Modeling Language (UML) Class [153], BPMN and ER diagrams. It is written in a extensible way that allows the addition more diagram types in the future.

To configure the type of the diagram, authors need to set the value of the type attribute as shown in the example of Listing A.23. Acceptable values are currently "uml", "bpmn" and "er".

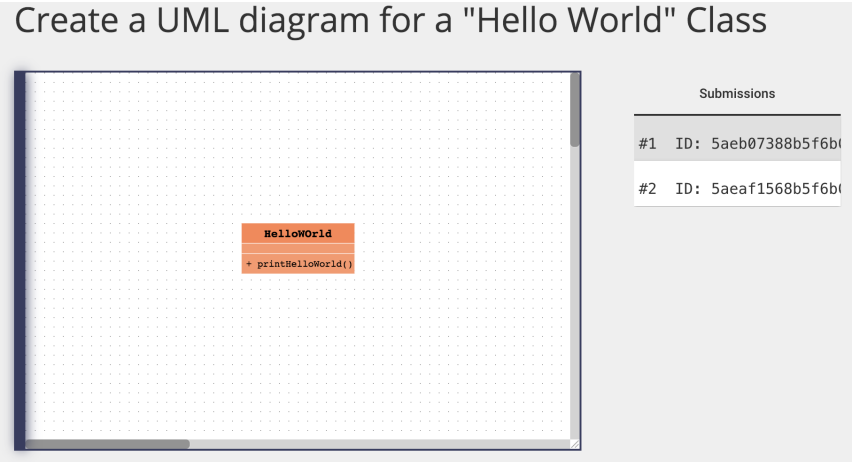
```
1 <asq-diagram-q id="asq-diagram-q" name="diagram0" type="uml" width="600"
      height="400">
2   <asq-stem><h3>Create a UML diagram for a "Hello World" Class</h3></asq-stem>
3 </asq-diagram-q>
```

Listing A.23. Example of an <asq-diagram-q> question type element for UML

Figure A.32 shows the interface of the <asq-diagram-q> question of Listing A.23. For both views, on the left there is a collapsible toolbar with all the tools necessary to create and edit a diagram. In our example the viewer (top of the figure) is editing the a Class diagram for a "HelloWorld" class with one *attribute* with the name "msg" and one *method* with the name "printMsg()". The presenter view (bottom) shows another submission from the list of submissions and has its toolbar collapsed.



Audience view



Presenter view

Figure A.32. Example of an `<asq-diagram-q>` question type element.

Appendix B

ASQ Software Architecture

B.1 Design and Implementation

ASQ follows a client server architecture and uses both acrshorthttp and the Websocket protocols for communication. An overview of the architecture is depicted in figure 4.1. In the back-end, an acrshorthttp server coupled with the business logic of the application serves static assets, bootstrap data (such as WebSocket connection configuration) and initial Hypertext Markup Language (HTML) which is first rendered server-side using a dynamic template framework. The logic is implement in JavaScript on top of the node.js runtime a choice mandated by the need for efficient I/O operations [176]. The Web Server is implemented using the express.js [116] framework. The routing scheme of the Web server design follows a RESTful [134] approach treating educational material (lectures, questions, answers) as Web resources.

B.1.1 **ASQ** Architectural Overview

User management in ASQ, involves user *roles* like presenters, teaching assistants and viewers and user permissions defined for each course. The core of the business logic is role-based, so presenters get a different main script than the viewers; and also view-based. For example presenters currently have two discrete views: the 'beamer' view which displays the presenter's version of the presentation and the 'cockpit' panel which displays feedback information like number of connected users, incoming answers, statistics, next and previous slides and more.

Subsequent client-server communication is implemented using the WebSocket [Fet11] protocol for reduced latency and higher throughput. This allows for real-time event collection that is crucial for monitoring progress, supporting compli-

cated assessment modes (for example peer assessment) or fine-grained logging of student actions. Interactions with the website that involve regular HTTP requests use cookie-based authentication. The same authentication is used for the WebSocket communication with an additional layer of role-based 'namespaces' [Rau14]. Different roles connect to different namespaces, which are pools of connection identifiers implemented by a software layer on top of WebSockets, albeit in the same host and port.

The persistence layer uses two different types of stores based on the volatility and access frequency of the stored data. A MongoDB database with collections for model data like question instances, presentations, sessions, users, and answers stores long term data. More volatile data that need frequent access like session events and socket events are stored in a Redis [39] key-value store for scalability and increased performance. Redis also provides a publish/subscribe implementation that helps scale the WebSocket component of the application.

`ASQ` supports both client- and server-side dynamic HTML content rendering with Dust.js [Wil18] templates.

B.1.2 Data Model

From a Domain Driven Design (DDD) [69] point of view, the *domain* of `ASQ` are Web-based interactive live presentations. The sub-domains are Authoring Presentations, Live Presentation Delivery, and Offline Analytics. The *model* of `ASQ` comprises:

User Represents a user of `ASQ`. Users can be `registered` or `guests`. If a user is a guest, they will automatically get a `screename`.

Presentation Represents an `ASQ` presentation. It stores fields like questions and exercises Unique Identifier (UID)s, type of presentation, number of slides, presentation settings, main file, owner, title and more.

Live session Represents a live presentation session. It holds references to the presenter, the slideshow and information about the active exercises and questions.

Exercise Represents an exercise and stores references to the exercise's questions.

Question Represents a question, storing information like its `type`, `stem`, `html`(markup) and `solution`.

Exercise Submission Keeps a record of each exercise submission. It does not contain the actual submission data of each question of the exercise.

Answer Represents a an answer to question. It stores the audience submission and references the associated question, session, exercise and user.

Assessment Assessment stores assessment scores for a specific answer as well as the user who performed the assessment(assessor) the user who is getting assessed(assessee) and a reference to the answer. It supports four modes *auto*, *manual*, *self* and *peer*. The auto mode denotes automatic assessment from `ASQ`. If the mode is manual then assessor is a presenter or a teaching assistant. In the other two modes the assessment is performed by students.

Session Event Represents a viewer or presenter event that happened during a live presentation. The full list of events is presented in section B.1.3.

Whitelist entry This model describes the privileges of a user for a specific live session. It holds a reference to the user and session.

Plugin Holds information about the status of a plugin in the system (installed/uninstalled, active/inactive).

Figure B.1 gives an overview of the relationship between the model entities.

B.1.3 Core Functionality

The backend logic follows a per-feature convention, where every feature is implemented in a dedicated directory inside `lib/`.

User Authentication and Authorization

Each client that requests an `ASQ` webpage through HTTP is assigned a user object, that corresponds to a user document in the database. Users can be “Registered” or “Guest”. `ASQ` uses cookie based authentication following the persistent session paradigm using the [pas18] library. Session identifiers are stored in the Redis database for increased performance and scalability. Using *[exp18]* we can

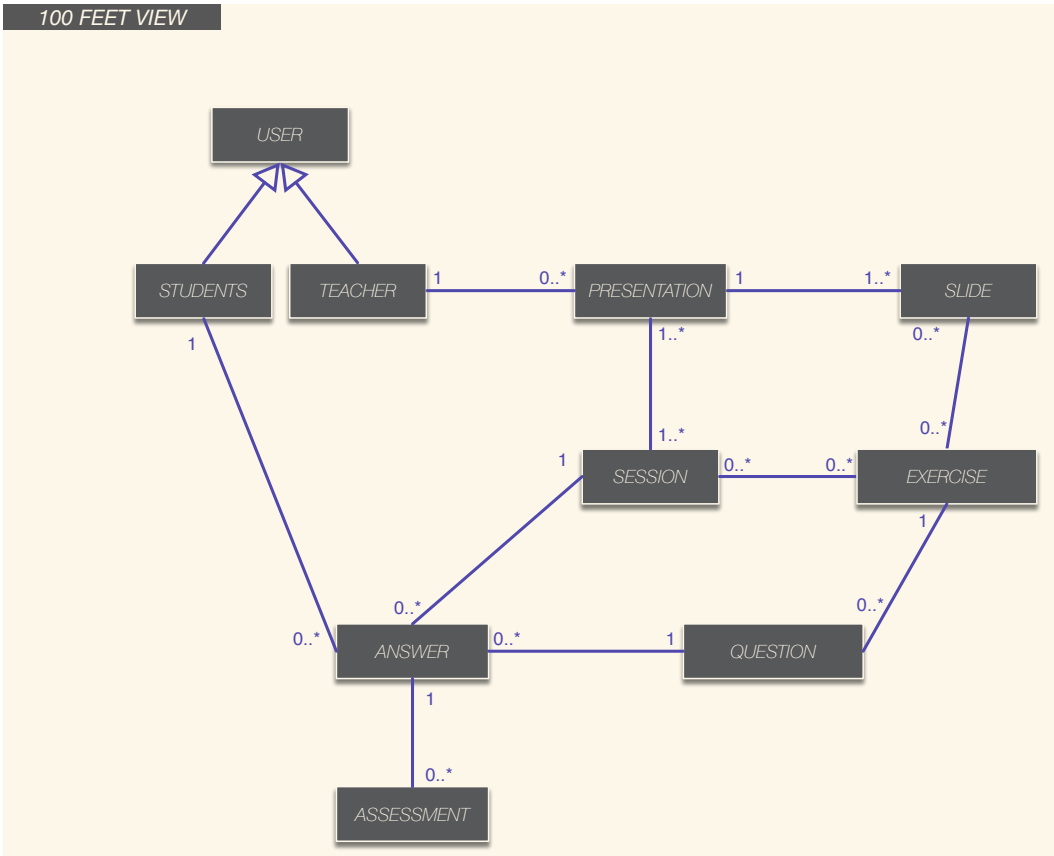


Figure B.1. Overview of the relationship between the entities of `ASQ`'s data model. For higher level overview see Figure 3.2.

set privilege requirements of each route. For example, the page that lists the presentation of a user should only be accessible by the this specific user. The majority of middleware functions are implemented under `lib/middleware`.

The ACL management for live presentations is implemented using the *whitelist entry* model. The user of every client in a live presentation session is associated with a role that grants specific privileges through a whitelist entry document. Currently `ASQ` supports four roles: *viewer*, *presenter*, *ghost* and *ta*. The viewer role represents audience members, i.e. students. Viewers can follow live presentations and answer quizzes. Presenters have a ctrl role and are the presentation owners the can advance slides and have access to submissions and analytics. The ghost role is used for the preview panes of the cockpit. They have similar privileges to viewers but their actions, for example submitting an answer, are not persisted in the system. ta, short for teaching assistants, have access to the cock-

pit where they can help manually assessing viewer submissions. Currently the ta role has the same privileges as the presenter. Websocket authentication is performed by sharing the HTTP authentication object of `passport.js` to identify a user and the whitelist entry model to determine user roles for a session. Once the authentication is complete, the requests are delegated to implementation logic of the corresponding feature.

Socket communication

Although the HTML and static assets of a live presentation are delivered through the HTTP, the rest of the communication between clients and server is performed through WebSockets. Examples of information that is exchanged through WebSockets are exercise submissions, slide advancing commands, statistics and state synchronization. The core of this logic is under `lib/sockets`.

Uploading a presentation

Files under `lib/upload` implement the business logic for uploading a presentation. Currently `ASQ` supports two file formats:a ZIP archive or a Portable Document Format (PDF) file. When one of the two types is uploaded `ASQ` creates a new entry in the presentations database.

ZIP archives need to contain at least one HTML file as a root file with an `.html` extension in its filename. `ASQ` will extract the ZIP archive in uniquely identified directory. It will then search the contents of the directoy for an HTML file. The first that matches (in alphabetical order) will be set as the main file of the presentation and will be parsed as explained in the next subsection.

PDF files will usually be presentations exported from a common presentation software such as PowerPoint or Keynote. `ASQ` will convert the PDF file to a *reveal.js* or *impress.js* HTML presentation using **slide2html**, a tool we developed that converts PDF documents to HTML presentations . Each page of the PDF will become a slide in the generated presentation. `ASQ` supports injecting questions in presentation generated from PDF files using an it `ASQ` URL. Authors need to include a question's UID following the schema of listing B.1 using the original presentation Software. `<hostname>` is the hostname of the `ASQ` server that the presentation is uploaded to. Currently, only questions residing in the same server as the presentation will be successufly resolved. The 24-character hexadecimal is the UID of the question in the `ASQ` database of the <hostname>. An example string that matches this pattern is `asq://asq.inf.usi.ch/questions/1234567890abcdef12345678`
After authors export the presentation to PDF and upload to `ASQ`, `ASQ` will parse

the presentation for a matching pattern as explained in the next subsection.

```
1 asq://<hostname>\\/questions\\/([0-9a-fA-F]{24}
```

Listing B.1. `ASQ` URL Schema definition (using a JavaScript regular expression) to identify questions for injection by their UID

Parsing a presentation

Once a presentation has been created, its main HTML file is parsed to extract settings, exercises and questions persist them in the database. The business logic for parsing is located in `lib/parsing`.

The parsing starts with searching for questions UIDs that match the Regular Expression of listing B.1. If there is match, `ASQ` queries the database to retrieve a question that matches the found UID. If an entry is found, the HTML markup of the question is injected in place of the matched string. The parsing starts with a sanity check to make sure that the data in the presentation are valid, such as that all exercises and questions include a UID, in the absence of which one will be created an injected.

Then a hook to parse the presentation settings is fired. One plugin that is subscribed to this hook is `asq-setting`. It will parse and persist the presentation-wide settings. Example settings include the slide flow type and the number of allowed submissions. When the hook execution is complete, the hook for plugins to parse the HTML of the presentation is fired. Plugins that respond to this hook will scan the presentation for their corresponding HTML elements, will parse them, and serialize them in the database. As a reminder, each hook callback can return a modified HTML string compared to the one it received as an argument. The main reasons for modifying the original markup are to remove confidential markup like ‘`<asq-solution> elements`, detect and correct mistakes – such as wrong syntax or missing required markup content– or inject additional content referenced via `ASQ` URLs.

Following the hook execution, the final step is to extract the UID of all the exercises and questions of a presentation as well as of the exercises per slide and questions per slide. This information will be added to the corresponding presentation entry in the database. Once the parsing is complete the presentation can be used for a live session.

Live presentation

The business logic for live presentations is located in `lib/live`, `lib/flow` and `lib/sockets` and `lib/submission`.

The process for starting a live session for a presentation starts with creating a new live session object (document in MongoDB lingo) that will be associated with the presentation.

Once the session is created, `ASQ` generates a whitelist entry for the presenter of the session that associates the user document of the owner of the presentation with the *role* "presenter" for this session. Whenever an audience member connects to the presentation a similiar whitelist entry is created that associated their user document with the role of "viewer" for the associated session.

The final step when starting a live presentation is to create an implicit audience question. Viewer can type messages intended for the presenter, such as comments or questions for the lectures as answers to this question. Submissions will be displayed in a dedicated view of the cockpit as "Viewers' questions".

Exercise Submission

When a viewer submits an exercise, the resulting exercise submission is forwarded from the WebSocket message handling logic to `/lib/submissions`.

The first step is to perform sanity checks: for example whether the question UIDs match the containing exercise UID or whether the user has exceeded the maximum allowed number of submissions for this specific exercise.

The next step is to save the exercise submission to the database and each individual question submission. We delegate this work the `asq-exercise` plugin and the pertinent question plugins. First `ASQ` will execute the `exercise_submission` hook to save the exercise submission to the database. Then for each question submission included in the exercise submission it will fire the `answer_submission` hook. Each question plugin performs further sanity checks to match the constraints of the specific question type and saves the answer to the database. Many plugins will then perform automatic assessment calculations and send the results back to the presenter and/or the viewers.

Exercise Assessment

`ASQ` support four types of assessment [238] as mentioned briefly in the *Assessment* model in subsection B.1.2:

1. **auto**. Some question types like multiple choice, highlight, text, order and buckets can be automatically assessed. This mode is automatically triggered

for compatible question types by ASQ which creates assessment entries in the database of the "auto" type and stores the automatically given grade.

2. **manual**. This is method allows instructors and teaching assistants to manually grade submissions. The submissions of students appear in the cockpit where teachers can quickly grade them as *correct* which corresponds to a score of 100 or *incorrect* which corresponds to a score of 0. The ability to specify a more fine grained score in the range of 0-100 will be added in the future.

3. **self**. In the self assessment mode students are provided with a rubric [11] that guides them through the steps to manually grade their assignment. They receive a score in the range 0-100.

4. **peer**. In this mode, once a student submits his answer to a question, ASQ will prompt them to assess the submission of a fellow classmate. The student follows the same rubric-based procedure describe in the self mode. Each student can assess more than one submission and their submission can be assessed multiple times as well.

The peer and self assessment modes need to be configured in the settings of the presentation via appropriate markup.

Real-time Analytics

In order to capture the interactions with the taught material, and to understand how they contribute to the learning process and student attention, for every live presentation ASQ tracks various events (e.g. a viewer connects to the ASQ presentation, a presenter advances a slide, a viewer submits an answer or is idle for a number of seconds). Depending on the nature of each event, it can be generated either in the user's browser of from the ASQ application server. Note, that we do not require users to login to ASQ, as long as a user's browser is connected to the ASQ presentation relevant events will be captured; closing the browser tab that contains the ASQ presentation will disconnect the user.

Generic Browser Events ASQ tracks some generic browser events like mouse movement, window focus and user input. These events are listed in Table B.1.

Table B.1. Overview of generic Web browser events monitored by ASQ.

Event Name	Description
tabhidden	The browser tab that displays the ASQ web app becomes hidden.
tabvisible	The browser tab that displays the ASQ web app becomes visible.
windowfocus	The browser window that displays the ASQ web app receives focus.
windowblur	The browser window that displays the ASQ web app loses focus (blurs in HTML terminology).
focusin, focus	An HTML element of the page received focus.
focusout, blur	An HTML element of the page receives focus.
click	A viewer clicks a pointing device inside the ASQ web page.
mousemove	A viewer moves a pointing device inside the ASQ web page.
dblclick	A viewer double-clicks a pointing device inside the ASQ web page.
contextmenu	A viewer right-clicks a pointing device inside the ASQ web page.
wheel	A viewer rotates the wheel button of a pointing device inside the ASQ web page.
touchstart	A viewer places a touch point touch point on the touch surface of the ASQ web page.
touchmove	A viewer moves s touch point along the touch surface of the ASQ web page.
touchend	A viewer removes a touch point touch point on the touch surface of the ASQ web page.
input	There is student input in the browser window that displays ASQ.
cut	A viewer performs a 'cut' operation in the browser window that displays ASQ.
copy	A viewer performs a 'copy' operation in the browser window that displays ASQ.
paste	A viewer performs a 'paste' operation in the browser window that displays ASQ.

ASQ-specific browser Events Events that are specific to the ASQ application are listed in Table B.2. These events include slide change events, input in ASQ question types and exercise operations (focus/edit/submit). These events may originate from a lower level browser event, for exampe `questioninput` may originate from an `input` event.

Table B.2. Overview of ASQ-specific Web browser events.

Event Name	Description
`slideenter`	The presentation in a viewer's ASQ web page advanced into a new slide.
`slideleave`	The presentation in a viewer's ASQ web page left the current slide.
`exercisefocus`	An ASQ exercise HTML element receives focus.
`exerciseblur`	An ASQ exercise HTML element blurs.
`input`	There is student input in the browser window that displays ASQ.
`questioninput`	Some ASQ question types emit this event when there is student input.
`exercisesubmit`	A viewer submits the solution to an ASQ exercise.
`answersubmit`	A viewer submits an answer for an ASQ question (an exercise can have multiple questions).
`exercise-edit`	A viewer clicked the edit button of an ASQ exercise.
`idle`	Emitted by the browser window that displays the ASQ web app when none of the above events has occurred for 10 seconds.

Server Generated Events The complete set of server generate events is listed in Table B.3.

Table B.3. Overview of server generated events of ASQ.

Event Name	Description
ctrl-connected	A user with the role of "presenter" connected to an ASQ presentation.
ctrl-disconnected	A user with the role of "presenter" disconnected from a live presentation.
ctrl-goto	A user with the role of "presenter" advanced a live presentation to a specific slide.
folo-connected	A user with the role of "viewer" connected to a live presentation.
folo-disconnected	A user with the role of "viewer" disconnected from a live presentation.
exercise-deactivated	An exercise inside a presentation has been deactivated.
exercise-activated	An exercise inside a presentation has been activated.
question-deactivated	A question inside a presentation has been deactivated.
question-activated	A question inside a presentation has been activated.

Bibliography

[1] Aas, P., Dixit, S., Eden, T., Lawson, B., Moon, S., Wu, X. and O'Hara, S. (eds) [2018]. *HTML 5.3*, editor's draft edn, W3C, chapter Elements.
URL: *http://w3c.github.io/html/*

[2] Abbott, M. L. and Fisher, M. T. [2009]. *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*, Pearson Education.

[3] Adams, E. N., Morrison, H. W. and Reddy, J. M. [1968]. Conversation with a computer as a technique of language instruction, *The Modern Language Journal* **52**(1): 3–16.

[4] Agarwal, P. K., Karpicke, J. D., Kang, S. H., Roediger III, H. L. and McDermott, K. B. [2008]. Examining the testing effect with open-and closed-book tests, *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition* **22**(7): 861–876.

[5] Aguilar-Roca, N. M., Williams, A. E. and O'Dowd, D. K. [2012]. The impact of laptop-free zones on student performance and attitudes in large lectures, *Computers & Education* **59**(4): 1300–1308.

[6] Alavi, H. S., Dillenbourg, P. and Kaplan, F. [2009]. Distributed awareness for class orchestration, *European Conference on Technology Enhanced Learning*, Springer, pp. 211–225.

[7] Albirini, A. [2007]. The crisis of educational technology, and the prospect of reinventing education, *Educational Technology & Society* **10**(1): 227–236.

[8] Allweyer, T. [2016]. *BPMN 2.0: introduction to the standard for business process modeling*, BoD–Books on Demand.

[9] Anderson, J. R. [1990]. *Cognitive psychology and its implications.*, WH Freeman/Times Books/Henry Holt & Co.

[10] Anderson, L. W., Krathwohl, D. R. and Bloom, B. S. [2001]. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*, Allyn & Bacon.

[11] Andrade, H. [2001]. The effects of instructional rubrics on learning to write, *Current Issues in Education* **4**(4).

[12] Arnold, K., Gosling, J. and Holmes, D. [2005]. *The Java programming language*, Addison Wesley Professional.

[13] Babad, E. Y. [1985]. Some correlates of teachers' expectancy bias, *American Educational Research Journal* **22**(2): 175–183.

[14] Baker, W. M., Lusk, E. J. and Neuhauser, K. L. [2012]. On the use of cell phones and other electronic devices in the classroom: Evidence from a survey of faculty and students, *Journal of Education for Business* **87**(5): 275–289.

[15] Bargh, J. A. and Chartrand, T. L. [1999]. The unbearable automaticity of being., *American psychologist* **54**(7): 462.

[16] Barrett, L. F., Tugade, M. M. and Engle, R. W. [2004]. Individual differences in working memory capacity and dual-process theories of the mind., *Psychological bulletin* **130**(4): 553.

[17] Bartlett, J. C. [1977]. Effects of immediate testing on delayed retrieval: Search and recovery operations with four types of cue., *Journal of Experimental Psychology: Human Learning and Memory* **3**(6): 719.

[18] Beilock, S. L. and Carr, T. H. [2001]. On the fragility of skilled performance: What governs choking under pressure?, *Journal of experimental psychology: General* **130**(4): 701.

[19] Beilock, S. L., Kulp, C. A., Holt, L. E. and Carr, T. H. [2004]. More on the fragility of performance: choking under pressure in mathematical problem solving., *Journal of Experimental Psychology: General* **133**(4): 584.

[20] Bessenoff, G. R. and Sherman, J. W. [2000]. Automatic and controlled components of prejudice toward fat people: Evaluation versus stereotype activation, *Social Cognition* **18**(4): 329–353.

[21] Bishop, J. L., Verleger, M. A. et al. [2013]. The flipped classroom: A survey of the research, *ASEE national conference proceedings, Atlanta, GA*, Vol. 30, pp. 1–18.

[22] Bixler, R., Blanchard, N., Garrison, L. and D'Mello, S. [2015]. Automatic detection of mind wandering during reading using gaze and physiology, *Proc. of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, pp. 299–306.

[23] Black, G. [2002]. A comparison of traditional, online, and hybrid methods of course delivery, *Journal of Business Administration Online* **1**(1): 1–9.

[24] *Blackboard Collaborate* [n.d.]. `https://www.blackboard.com/platforms/collaborate/overview.aspx`.
URL: *https://www.blackboard.com/platforms/collaborate/overview.aspx*

[25] Blackwell, A., McLean, A., Noble, J. and Julian, R. [2013]. Collaboration and learning through live coding, *Technical report*, Dagstuhl.

[26] Blair, A. M. [2008]. Textbooks and methods of note-taking in early modern europe, *Scholarly knowledge: textbooks in early modern Europe* .

[27] Blair, I. V. and Banaji, M. R. [1996]. Automatic and controlled processes in stereotype priming., *Journal of personality and social psychology* **70**(6): 1142.

[28] Bloom, B. S. [1984]. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring., *Educational Researcher* **13**(6): 4–16.
URL: *http://www.eric.ed.gov/ERICWebPortal/detail?accno=EJ303699*

[29] Bonwell, C. C. and Eison, J. A. [1991]. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.*, ERIC.

[30] Bostock, M. et al. [2012]. D3. js, *Data Driven Documents* **492**: 701.

[31] Botvinick, M. M., Cohen, J. D. and Carter, C. S. [2004]. Conflict monitoring and anterior cingulate cortex: an update, *Trends in cognitive sciences* **8**(12): 539–546.

[32] Bowen, W. G., Chingos, M. M., Lack, K. A. and Nygren, T. I. [2012]. Interactive learning online at public universities: Evidence from randomized trials.

[33] Brooke, J. et al. [1996]. Sus-a quick and dirty usability scale, *Usability evaluation in industry* **189**(194): 4–7.

[34] Buettner, R. [2013]. Cognitive workload of humans using artificial intelligence systems: towards objective measurement applying eye-tracking technology, *Annual Conference on Artificial Intelligence*, Springer, pp. 37–48.

[35] Bunce, D. M., Flens, E. A. and Neiles, K. Y. [2010]. How long can students pay attention in class? a study of student attention decline using clickers, *Journal of Chemical Education* **87**(12): 1438–1443.

[36] Burke, L. A. and Ray, R. [2008]. Re-setting the concentration levels of students in higher education: an exploratory study, *Teaching in Higher Ed.* **13**(5): 571–582.

[37] Campbell, D. T. and Stanley, J. C. [2015]. *Experimental and quasi-experimental designs for research*, Ravenio Books.

[38] Campbell, J. P., DeBlois, P. B. and Oblinger, D. G. [2007]. Academic analytics: A new tool for a new era, *EDUCAUSE review* **42**(4): 40.

[39] Carlson, J. L. [2013]. *Redis in action*, Manning Publications Co.

[40] Carre, C. [1993]. The first year of teaching, *Learning to teach* pp. 191–211.

[41] Carrier, M. and Pashler, H. [1992]. The influence of retrieval on retention, *Memory & Cognition* **20**(6): 633–642.

[42] Chandler, P. and Sweller, J. [1991]. Cognitive load theory and the format of instruction, *Cognition and instruction* **8**(4): 293–332.

[43] Chapman, G. B. and Johnson, E. J. [2002]. Incorporating the irrelevant: Anchors in judgments of belief and value, *Heuristics and biases: The psychology of intuitive judgment* pp. 120–138.

[44] Chen, P. P.-S. [1988]. The entity-relationship model—toward a unified view of data, *Readings in artificial intelligence and databases*, Elsevier, pp. 98–111.

[45] Chickering, A. W. and Gamson, Z. F. [1987]. Seven principles for good practice in undergraduate education., *AAHE bulletin* **3**: 7.

[46] Clark, R. E. [2001a]. *Learning from media: Arguments, analysis, and evidence*, Vol. 1, IAP.

[47] Clark, R. E. [2001b]. *New Directions: Cognitive and Motivational Research Issues*, 1st edn, Information Age Publishing, chapter 15, pp. 263–298.

[48] Colella, V. [2000]. Participatory simulations: Building collaborative understanding through immersive dynamic modeling, *The journal of the Learning Sciences* **9**(4): 471–500.

[49] Cooper, S. and Sahami, M. [2013]. Reflections on stanford's moocs, *Commun. ACM* **56**(2): 28–30.
URL: *http://doi.acm.org/10.1145/2408776.2408787*

[50] Darley, C. F. and Murdock, B. B. [1971]. Effects of prior free recall testing on final recall and recognition., *Journal of Experimental Psychology* **91**(1): 66.

[51] Date, C. J. and Darwen, H. [1989]. *A guide to the SQL Standard: a user's guide to the standard relational language SQL*, Addison-Wesley.

[52] Davis, S. [2003]. Observations in classrooms using a network of handheld devices, *Journal of Computer Assisted Learning* **19**(3): 298–307.
URL: *http://dx.doi.org/10.1046/j.0266-4909.2003.00031.x*

[53] de Ridder-Symoens, H. and Rüegg, W. [1992]. *Universities in Early Modern Europe, 500-1800*, Cambridge University Press.

[54] Devine, P. G. and Monteith, M. J. [1999]. Automaticity and control in stereotyping, *in* S. Chaiken and Y. Trope (eds), *Dual-process theories in social psychology*, Guilford Press, New York, NY, US, pp. 339–360.

[55] Devine, P. G., Plant, E. A., Amodio, D. M., Harmon-Jones, E. and Vance, S. L. [2002]. The regulation of explicit and implicit race bias: the role of motivations to respond without prejudice., *Journal of personality and social psychology* **82**(5): 835.

[56] Dillenbourg, P. [2013]. Design for classroom orchestration, *Computers & Education* **69**: 485–492.

[57] Dimitri, G. [2014]. Custom elements, *W3c working draft*, W3C. http://www.w3.org/TR/2014/WD-custom-elements-20141216/.

[58] Dimitri, G. and Hajime, M. [2014]. Html imports, *W3c working draft*, W3C. http://www.w3.org/TR/2014/WD-html-imports-20140311/.

[59] Dimitri, G. and Ito, H. [2014]. Shadow dom, *W3c working draft*, W3C. http://www.w3.org/TR/2014/WD-shadow-dom-20140617/.

[60] Domenic, D. [2018]. Custom elements, *W3c editor's draft*, W3C. https://w3c.github.io/webcomponents/spec/custom/.

[61] Dougiamas, M. [2004]. Moodle: A virtual learning environment for the rest of us, *TESL-EJ* **8**(2): 1–8.

[62] Draper, S. W. and Brown, M. I. [2004]. Increasing interactivity in lectures using an electronic voting system., *J. Comp. Assisted Learning* **20**(2): 81–94. **URL:** *http://dblp.uni-trier.de/db/journals/jcal/jcal20.html#DraperB04*

[63] Duchowski, A. T. [2007]. Eye tracking methodology, *Theory and practice* **328**.

[64] Dufresne, R. J., Gerace, W. J., Leonard, W. J., Mestre, J. P. and Wenk, L. [1996]. Classtalk: A classroom communication system for active learning, *Journal of computing in higher education* **7**(2): 3–47.

[65] Duncan, D. [2005]. *Clickers in the Classroom: How to Enhance Science Teaching Using Classroom Response Systems*, Pearson Education, San Francisco.

[66] Earle, R. S. [2002]. The integration of instructional technology into public education: Promises and challenges, *EDUCATIONAL TECHNOLOGY-SADDLE BROOK THEN ENGLEWOOD CLIFFS NJ-* **42**(1): 5–13.

[67] ECMA [2015]. Draft specification for es.next (ecma-262 edition 6), *Ecmascript working draft*, ECMA.

[68] Ertmer, P. A. and Ottenbreit-Leftwich, A. [2013]. Removing obstacles to the pedagogical changes required by jonassen's vision of authentic technology-enabled learning, *Computers & Education* **64**: 175–182.

[69] Evans, E. [2004]. *Domain-driven design: tackling complexity in the heart of software*, Addison-Wesley Professional.

[70] Fazio, R. H., Jackson, J. R., Dunton, B. C. and Williams, C. J. [1995]. Variability in automatic activation as an unobtrusive measure of racial attitudes: a bona fide pipeline?, *Journal of personality and social psychology* **69**(6): 1013.

[71] Fazio, R., Sanbonmatsu, D., Powell, M. and Kardes, F. [1986]. On the automatic activation of attitudes, *Journal of Personality and Social Psychology* **50**(2): 229–238.

[72] Feldon, D. F. [2007]. Cognitive load and classroom teaching: The double-edged sword of automaticity, *Educational Psychologist* **42**(3): 123–137.

[73] Fischer, F. and Dillenbourg, P. [2006]. Challenges of orchestrating computer-supported collaborative learning., *87th annual meeting of the American Educational Research Association (AERA)* .

[74] Freeman, M. and Blayney, P. [2005]. Promoting interactive in-class learning environments: A comparison of an electronic response system with a traditional alternative, *Innovation* .

[75] Freeman, M. J. and Ullman, C. [2008]. Digital interactive system for providing full interactivity with live programming events. US Patent 7,448,063.

[76] Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H. and Wenderoth, M. P. [2014]. Active learning increases student performance in science, engineering, and mathematics, *Proceedings of the National Academy of Sciences* **111**(23): 8410–8415.

[77] Fried, C. B. [2008]. In-class laptop use and its effects on student learning, *Computers & Education* **50**(3): 906–914.

[78] Friesen, N. [2014]. A brief history of the lecture: A multi-media analysis, *MedienPädagogik: Zeitschrift für Theorie und Praxis der Medienbildung* **24**: 136–153.

[79] Fullan, M. [2000]. The three stories of education reform, *Phi Delta Kappan* **81**(8): 581–584.

[80] Fullan, M. [2013]. *Stratosphere: Integrating technology, pedagogy, and change knowledge*, Pearson Don Mills, Canada.

[81] Gauci, S. A., Dantas, A. M., Williams, D. A. and Kemm, R. E. [2009]. Promoting student-centered active learning in lectures with a personal response system, *Advances in Physiology Education* **33**(1): 60–71.

[82] Gibbs, G. and Jenkins, A. [1984]. Break up your lectures: or christaller sliced up, *Journal of Geography in Higher Education* **8**(1): 27–39.

[83] Glover, J. A. [1989]. The" testing" phenomenon: Not gone but nearly forgotten., *Journal of Educational Psychology* **81**(3): 392.

[84] Goldinger, S. D., Kleider, H. M., Azuma, T. and Beike, D. R. [2003]. "blaming the victim" under memory load, *Psychological Science* **14**(1): 81–85.

[85] Good, T. L. and Nichols, S. L. [2001]. Expectancy effects in the classroom: A special focus on improving the reading performance of minority students in first-grade classrooms, *Educational Psychologist* **36**(2): 113–126.

[86] Gray, R. [2004]. Attending to the execution of a complex sensorimotor skill: expertise differences, choking, and slumps., *Journal of Experimental Psychology: Applied* **10**(1): 42.

[87] Greenwald, A. G. and Banaji, M. R. [1995]. Implicit social cognition: attitudes, self-esteem, and stereotypes., *Psychological review* **102**(1): 4.

[88] Hanawalt, N. G. and Tarr, A. G. [1961]. The effect of recall upon recognition., *Journal of Experimental Psychology* **62**(4): 361.

[89] Hart, A. [1987]. The political economy of interactive video in british higher education, *Interactive media: working methods and practical applications*, Halsted Press, pp. 171–189.

[90] Hart, S. G. and Staveland, L. E. [1988]. Development of nasa-tlx (task load index): Results of empirical and theoretical research, *Advances in psychology*, Vol. 52, Elsevier, pp. 139–183.

[91] Hauswirth, M. and Adamoli, A. [2009]. Solve and evaluate with informa: a java-based classroom response system for teaching java, *Proc. of PPPJ '09*, pp. 1–10.

[92] Hauswirth, M. and Adamoli, A. [2013]. Teaching java programming with the informa clicker system, *Sci. Comput. Program.* **78**(5): 499–520.
URL: *http://dx.doi.org/10.1016/j.scico.2011.06.006*

[93] Hembrooke, H. and Gay, G. [2003]. The laptop and the lecture: The effects of multitasking in learning environments, *Journal of computing in higher education* **15**(1): 46–64.

[94] Hogan, R. M. and Kintsch, W. [1971]. Differential effects of study and test trials on long-term recognition and recall, *Journal of Verbal Learning and Verbal Behavior* **10**(5): 562–567.

[95] Hucko, M., Gaspar, P., Pikuliak, M., Triglianos, V., Pautasso, C. and Bielikova, M. [n.d.]. Short texts analysis for teacher assistance during live interactive classroom presentations(to appear), *Proceedings of the World Symposium on Digital Intelligence for Systems and Machines (DISA*, Koŝice, Slovakia.

[96] Hung, D. [2002]. Situated cognition and problem-based learning: Implications for learning and instruction with technology, *Journal of Interactive Learning Research* **13**(4): 393–414.

[97] Hwang, S. Y. and Kim, M. J. [2006]. A comparison of problem-based learning and lecture-based learning in an adult health nursing course, *Nurse education today* **26**(4): 315–321.

[98] Illich, I. and Lang, A. [1973]. Tools for conviviality.

[99] *Introduction to Web Components* [n.d.]. https://www.webcomponents.org/introduction.
URL: *https://www.webcomponents.org/introduction*

[100] Jazayeri, M. [2004]. The education of a software engineer, *Proceedings of the 19th IEEE international conference on Automated software engineering*, IEEE, pp. 18–xxvii.

[101] Jazayeri, M. [2015]. Combining mastery learning with project-based learning in a first programming course: an experience report, *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, IEEE, pp. 315–318.

[102] Johnson, C. I. and Mayer, R. E. [2009]. A testing effect with multimedia learning., *Journal of Educational Psychology* **101**(3): 621.

[103] Johnstone, A. H. and Percival, F. [1976]. Attention breaks in lectures., *Education in chemistry* **13**(2): 49–50.

[104] Judson, E. and Sawada, D. [2006]. Audience response systems: Insipid contrivances or inspiring tools?, *in* D. A. Banks (ed.), *Audience Response Systems in Higher Education: Applications and Cases*, IGI Global, chapter 2, pp. 26–39.

[105] Kagan, D. M. [1992]. Professional growth among preservice and beginning teachers, *Review of educational research* **62**(2): 129–169.

[106] Kang, S. H., McDermott, K. B. and Roediger III, H. L. [2007]. Test format and corrective feedback modify the effect of testing on long-term retention, *European Journal of Cognitive Psychology* **19**(4-5): 528–558.

[107] Kaput, J. J. and Hegedus, S. J. [2002]. Exploiting classroom connectivity by aggregating student constructions to create new learning opportunities, *26th Conference of the International Group for the Psychology of Mathematics Education*.

[108] Kato, J. and Goto, M. [2016]. Live tuning: Expanding live programming benefits to non-programmers, *Proceedings of the Second Workshop on Live Programming Systems (ECOOP LIVE'16)*, Vol. 6.

[109] Kay, R. H. and Lauricella, S. [2011]. Exploring the benefits and challenges of using laptop computers in higher education classrooms: A formative analysis, *Canadian Journal of Learning and Technology/La revue canadienne de l'apprentissage et de la technologie* **37**(1).

[110] Leinhardt, G. and Greeno, J. G. [1986]. The cognitive skill of teaching., *Journal of educational psychology* **78**(2): 75.

[111] Lindquist, S. I. and McLean, J. P. [2011]. Daydreaming and its correlates in an educational environment, *Learning and Individual Differences* **21**(2): 158–167.

[112] Lortie, D. C. [1975]. Schoolteacher, a sociological study. by dan c. lortie. chicago, ill.: University of chicago, 1975, *NASSP Bulletin* **59**(394): 119–120.

[113] Lowman, J. and Lowman, J. [1984]. *Mastering the techniques of teaching*, Vol. 1990, Jossey-Bass San Francisco.

[114] Maass, J. K. and Pavlik Jr, P. I. [2016]. Modeling the influence of format and depth during effortful retrieval practice, *Proceedings of the 9th International Conference on Educational Data Mining*, Proceedings of the 9th International Conference on Educational Data Mining, pp. 143–150.

[115] Macrae, C. N., Hewstone, M. and Griffiths, R. J. [1993]. Processing load and memory for stereotype-based information, *European Journal of Social Psychology* **23**(1): 77–87.

[116] Mardan, A. [2014]. *Express.js Guide: The Comprehensive Book on Express.js*, Azat Mardan.

[117] Mayer, E. R. [2005]. *Cognitive Theory of Multimedia Learning*, Cambridge University Press, pp. 31–48.

[118] Mayer, R. E. and Moreno, R. [1998]. A cognitive theory of multimedia learning: Implications for design principles, *Journal of Educational Psychology* **91**(2): 358–368.

[119] Mayer, R. E., Stull, A., DeLeeuw, K., Almeroth, K., Bimber, B., Chun, D., Bulger, M., Campbell, J., Knight, A. and Zhang, H. [2009]. Clickers in college classrooms: Fostering learning with questioning methods in large lecture classes, *Contemporary educational psychology* **34**(1): 51–57.

[120] McDaniel, M. A., Anderson, J. L., Derbish, M. H. and Morrisette, N. [2007]. Testing the testing effect in the classroom, *European Journal of Cognitive Psychology* **19**(4-5): 494–513.

[121] McDaniel, M. A., Kowitz, M. D. and Dunay, P. K. [1989]. Altering memory through recall: The effects of cue-guided retrieval processing, *Memory & Cognition* **17**(4): 423–434.

[122] McDaniel, M. A. and Masson, M. E. [1985]. Altering memory representations through retrieval., *Journal of Experimental Psychology: Learning, Memory, and Cognition* **11**(2): 371.

[123] Merkel, D. [2014]. Docker: lightweight linux containers for consistent development and deployment, *Linux Journal* **2014**(239): 2.

[124] Misseyanni, A., Marouli, C., Papadopoulou, P., Lytras, M. and Gastardo, M. T. [2016]. Stories of active learning in stem: Lessons for stem education, *Proceedings of the International Conference The Future of Education,*, p. 232À236.

[125] Moors, A. and De Houwer, J. [2006]. Automaticity: a theoretical and conceptual analysis., *Psychological bulletin* **132**(2): 297.

[126] Moreau-Mathis, J. [2016]. *Babylon. js Essentials*, Packt Publishing Ltd.

[127] Morris, C. D., Bransford, J. D. and Franks, J. J. [1977]. Levels of processing versus transfer appropriate processing, *Journal of verbal learning and verbal behavior* **16**(5): 519–533.

[128] Nettle, E. [1998]. Stability and change in the beliefs of student teachers during practice teaching, *Teaching and Teacher Education* **14**(2): 193–204.

[129] Nguyen, A., Piech, C., Huang, J. and Guibas, L. [2014]. Codewebs: Scalable homework search for massive open online programming courses, *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, ACM, New York, NY, USA, pp. 491–502.

[130] Nicol, D. J. and Boyle, J. T. [2003]. Peer instruction versus class-wide discussion in large classes: a comparison of two interaction methods in the wired classroom, *Studies in Higher Education* **28**(4): 457–473.

[131] Ohlsson, S. [1996]. Learning from performance errors., *Psychological Review* **103**(2): 241–262.

[132] Owens, M. and Allen, G. [2010]. *SQLite*, Springer.

[133] Paul, R. [1991]. *Teaching strategies for effective learning*, second edn, RoutledgeFalmer, chapter 9, pp. 145–175.

[134] Pautasso, C., Zimmermann, O. and Leymann, F. [2008]. Restful web services vs. big'web services: making the right architectural decision, *Proceedings of the 17th international conference on World Wide Web*, ACM, pp. 805–814.

[135] Payne, B. K., Lambert, A. J. and Jacoby, L. L. [2002]. Best laid plans: Effects of goals on accessibility bias and cognitive control in race-based misperceptions of weapons, *Journal of Experimental Social Psychology* **38**(4): 384–396.

[136] Penades, S. [2015]. An Introduction to Web Components, *Web Components London*, webcomponents.org.

[137] Penuel, W., Roschelle, J. and Abrahamson, L. [2005]. Research on classroom networks for whole-class activities, *Wireless and Mobile Technologies in Education, 2005. WMTE 2005. IEEE International Workshop on*, pp. 8 pp.–.

[138] Persico, D. and Pozzi, F. [2015]. Informing learning design with learning analytics to improve teacher inquiry, *British Journal of Educational Technology* **46**(2): 230–248.

[139] Poulis, J., Massen, C., Robens, E. and Gilbert, M. [1998]. Physics lecturing with audience paced feedback, *American Journal of Physics* **66**: 439–441.

[140] Prieto, L. P., Wen, Y., Caballero, D., Sharma, K. and Dillenbourg, P. [2014]. Studying teacher cognitive load in multi-tabletop classrooms using mobile eye-tracking, *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ACM, pp. 339–344.

[141] Prieto Santos, L. P. et al. [2016]. Teaching analytics: Towards automatic extraction of orchestration graphs using wearable sensors, *Int'l Learning Analytics and Knowledge*, number EPFL-CONF-216918.

[142] Prince, M. [2004]. Does active learning work? a review of the research, *Journal of engineering education* **93**(3): 223–231.

[143] Rangrej, A., Kulkarni, S. and Tendulkar, A. V. [2011]. Comparative study of clustering techniques for short text documents, *Proceedings of the 20th international conference companion on World wide web*, ACM, pp. 111–112.

[144] Ravizza, S. M., Uitvlugt, M. G. and Fenn, K. M. [2016]. Logged in and zoned out: How laptop internet use relates to classroom learning, *Psychological Science* p. 0956797616677314.

[145] Rich, Y. [1990]. Ideological impediments to instructional innovation: The case of cooperative learning, *Teaching and Teacher Education* **6**(1): 81–91.

[146] Risko, E. F., Anderson, N., Sarwal, A., Engelhardt, M. and Kingstone, A. [2012]. Everyday attention: variation in mind wandering and memory in a lecture, *Applied Cognitive Psychology* **26**(2): 234–242.

[147] Risko, E. F., Buchanan, D., Medimorec, S. and Kingstone, A. [2013]. Everyday attention: Mind wandering and computer use during lectures, *Computers & Education* **68**: 275–283.

[148] Ritts, V., Patterson, M. L. and Tubbs, M. E. [1992]. Expectations, impressions, and judgments of physically attractive students: A review, *Review of educational research* **62**(4): 413–426.

[149] Roediger, H. L. and Butler, A. C. [2011]. The critical role of retrieval practice in long-term retention, *Trends in cognitive sciences* **15**(1): 20–27.

[150] Roediger III, H. L. and Karpicke, J. D. [2006]. The power of testing memory: Basic research and implications for educational practice, *Perspectives on Psychological Science* **1**(3): 181–210.

[151] Rollag, K. and Billsberry, J. [2012]. Technology as the enabler of a new wave of active learning.

[152] Rosenthal, R. [2002]. The pygmalion effect and its mediating mechanisms, *Improving academic achievement*, Elsevier, pp. 25–36.

[153] Rumbaugh, J., Jacobson, I. and Booch, G. [2004]. *Unified Modeling Language Reference Manual, The (2Nd Edition)*, Pearson Higher Education.

[154] Safari, M., Yazdanpanah, B., Ghafarian, H. R. and Yazdanpanah, S. [2006]. Comparing the effect of lecture and discussion methods on studentslearning and satisfaction, *Iranian journal of medical education* **6**(1): 59–64.

[155] Sana, F., Weston, T. and Cepeda, N. J. [2013]. Laptop multitasking hinders classroom learning for both users and nearby peers, *Computers & Education* **62**: 24–31.

[156] Sauro, J. [2011]. *A practical guide to the system usability scale: Background, benchmarks & best practices*, Measuring Usability LLC Denver, CO.

[157] Scerbo, M. W., Warm, J. S., Dember, W. N. and Grasha, A. F. [1992]. The role of time and cuing in a college lecture, *Contemp. Educational Psychology* **17**(4): 312–328.

[158] Schneider, W. and Chein, J. M. [2003]. Controlled & automatic processing: behavior, theory, and biological mechanisms, *Cognitive Science* **27**(3): 525–559.

[159] Scouller, K. [1998]. The influence of assessment method on students' learning approaches: Multiple choice question examination versus assignment essay, *Higher Education* **35**(4): 453–472.

[160] Shrestha, P., Jacquin, C. and Daille, B. [2012]. Clustering short text and its evaluation, *Computational Linguistics and Intelligent Text Processing* pp. 169–180.

[161] Srba, I. and Bielikova, M. [2015]. Askalot: community question answering as a means for knowledge sharing in an educational organization, *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, ACM, pp. 179–182.

[162] Stuart, J. and Rutherford, R. [1978]. Medical student concentration during lectures, *The lancet* **312**(8088): 514–516.

[163] Studio, T. [2008]. 1.2 user manual, *Tobii Technology AB* .

[164] Sun, H. J., Huang, M. X., Ngai, G. and Chan, S. C. F. [2014]. Nonintrusive multimodal attention detection, *ACHI 2014, The Seventh International Conference on Advances in Computer-Human Interactions*, Citeseer, pp. 192–199.

[165] Sun, R., Slusarz, P. and Terry, C. [2005]. The interaction of the explicit and the implicit in skill learning: a dual-process approach., *Psychological review* **112**(1): 159.

[166] Sweller, J. [1988]. Cognitive load during problem solving: Effects on learning, *Cognitive science* **12**(2): 257–285.

[167] Sweller, J. [1989]. Cognitive technology: Some procedures for facilitating learning and problem solving in mathematics and science., *Journal of Educational Psychology* **81**(4): 457–466.

[168] Sweller, J. [1999]. *Instructional Design in Technical Areas. Australian Education Review, No. 43.*, ERIC.

[169] Sweller, J., Chandler, P., Tierney, P. and Cooper, M. [1990]. Cognitive load as a factor in the structuring of technical material., *Journal of Experimental Psychology: General* **119**(2): 176.

[170] Szafir, D. and Mutlu, B. [2012]. Pay attention!: Designing adaptive agents that monitor and improve user engagement, *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 11–20.

[171] Szafir, D. and Mutlu, B. [2013]. Artful: Adaptive review technology for flipped learning, *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*.

[172] Szpunar, K. K., Khan, N. Y. and Schacter, D. L. [2013]. Interpolated memory tests reduce mind wandering and improve learning of online lectures, *Proceedings of the National Academy of Sciences* **110**(16): 6313–6317.

[173] Szpunar, K. K., Moulton, S. T. and Schacter, D. L. [2013]. Mind wandering and education: from the classroom to online learning.

[174] Thompson, A. D., Simonson, M. R. and Hargrave, C. P. [1992]. *Educational technology: A review of the research*, Association for Educational Communications and Technology.

[175] Thompson, S. C., Kyle, D., Osgood, A., Quist, R. M., Phillips, D. J. and McClure, M. [2004]. Illusory control and motives for control: The role of connection and intentionality, *Motivation and Emotion* **28**(4): 315–330.

[176] Tilkov, S. and Vinoski, S. [2010]. Node.js: Using JavaScript to Build High-Performance Network Programs, *IEEE Internet Computing* **14**(6): 80–83.

[177] Triglianos, V., Praharaj, S., Pautasso, C., Bozzon, A. and Hauff, C. [2017]. Measuring student behaviour dynamics in a large interactive classroom setting, *25th International Conference on User Modelling, Adaption and Personalisation (UMAP 2017)*, ACM, ACM, Bratislava, Slovakia, pp. 212–220.
URL: *http://dl.acm.org/authorize?N31530*

[178] Trollip, S. R. and Alessi, S. M. [2001]. Multimedia for learning: methods and development.

[179] Valdez, G., McNabb, M., Foertsch, M., Anderson, M., Hawkes, M. and Raack, L. [1999]. *Computer-based technology and learning: Evolving uses and expectations.*, ERIC, Oak Brook, IL.

[180] Walton, P. [2014]. Web components and the future of CS, *Proc. of SFHTML5*. `http://webcomponents.org/presentations/web-components-and-the-future-of-css/`.

[181] Wegner, D. M. [1994]. Ironic processes of mental control., *Psychological review* **101**(1): 34.

[182] Weinstein, Y., Nunes, L. D. and Karpicke, J. D. [2016]. On the placement of practice questions during study., *Journal of Experimental Psychology: Applied* **22**(1): 72.

[183] Wheatley, T. P. and Wegner, D. M. [2001]. *Automaticity in action*, Pergamon, London, pp. 991–993.

[184] Whitten II, W. B. and Bjork, R. A. [1977]. Learning from tests: Effects of spacing, *Journal of Verbal Learning and Verbal Behavior* **16**(4): 465–478.

[185] Wideen, M., Mayer-Smith, J. and Moon, B. [1998]. A critical analysis of the research on learning to teach: Making the case for an ecological perspective on inquiry, *Review of educational research* **68**(2): 130–178.

[186] Widom, J. [2012]. From 100 students to 100,000, `http://wp.sigmod.org/?p=165`.
URL: *http://wp.sigmod.org/?p=165*

[187] Wilson, K. and Korn, J. H. [2007]. Attention during lectures: Beyond ten minutes, *Teaching of Psychology* **34**(2): 85–89.

[188] Wood, E., Zivcakova, L., Gentile, P., Archer, K., De Pasquale, D. and Nosko, A. [2012]. Examining the impact of off-task multi-tasking with technology on real-time classroom learning, *Computers & Education* **58**(1): 365–374.

[189] Wyatt, D. H. [1982]. Applying pedagogical principles to call courseware development.

[190] Yin, H., Moghadam, J. and Fox, A. [2015]. Clustering student programming assignments to multiply instructor leverage, *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, ACM, Vancouver, BC, Canada, pp. 367–372.

[191] Zakai, A. [2011]. Emscripten: an llvm-to-javascript compiler, *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, ACM, pp. 301–312.

Web References

[ace18] Ace Editor, 2018.

[BÍ15] J.R. Bédard. Isomorphic javascript. http://isomorphic.net/, 2015.

[BGSC18] Kent Beck, Erich Gamma, David Saff, and Mike Clark. JUnit. https://junit.org/junit5/, 2018.

[bla] Blackboard collaborate. http://www.blackboard.com/online-collaborative-learning/blackboard-collaborate.html.

[C+10] Ricardo Cabello et al. Three. js. https://github.com/mrdoob/three.js, 2010.

[c915] c9. architect. https://github.com/c9/architect, 2015.

[cuj15] cujoJS. wire. https://github.com/cujojs/wire, 2015.

[Dow15] Nick Downie. Chart. js: Open source html5 charts for your website. https://www.chartjs.org/, 2015.

[EH11] Hakim El Hattab. reveal.js. https://revealjs.com/, 2011.

[eth] Glisser. https://www.ethz.ch/en/the-eth-zurich/education/innovation/eduapp.html.

[exp18] express.js middleware. https://expressjs.com/en/guide/using-middleware.html, 2018.

[Fet11] Ian Fette. The websocket protocol. https://tools.ietf.org/html/rfc6455/, 2011.

[gli] Glisser. https://www.glisser.com/.

[goo] Google forms. https://www.google.com/forms/about/.

[Hoo15] H Hoodie. Hoodie Homepage. http://hood.ie/, 2015. Accessed: 2015-02-25.

[Jac15] Roy Jacobs. intravenous. https://github.com/RoyJacobs/intravenous, 2015.

[kan09] kangax. Detecting global variable leaks. http://perfectionkills.com/detecting-global-variable-leaks/, 2009.

[key] Keynote. https://www.apple.com/lae/keynote//.

[Kun16] G Kunz. Chartist-simple responsive charts. http://gionkunz.github.io/chartist-js/index.html, 2016.

[MBJ+11] Matt Mullenweg, R Boren, M Jaquith, A Ozz, and P Westwood. Wordpress. https://wordpress.org/, 2011.

[Mdd15] Moodle Moodle developer documentation. Event 2. https://docs.moodle.org/dev/Event_2, 2015. Accessed: 2015-02-25.

[mdn18a] Css selectors. https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors, 2018.

[mdn18b] Dom click event. https://developer.mozilla.org/en-US/docs/Web/Events/click, 2018.

[mdn18c] Dom input event. https://developer.mozilla.org/en-US/docs/Web/Events/input, 2018.

[mdn18d] Http headers. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers, 2018.

[nea] Nearpod. https://nearpod.com/.

[pas18] passport.js. http://www.passportjs.org/, 2018.

[pia] Piazza. https://piazza.com/.

[Pol18] p Polymer. Polymer Homepage. https://www.polymer-project.org/, 2018.

[pow] Powerpoint. https://products.office.com/en/powerpoint.

[pre] Prezi. https://prezi.com/.

[pro] Proprofs. https://www.proprofs.com/.

[Rau14] Guillermo Rauch. Rooms and Namespaces. http://socket.io/docs/rooms-and-namespaces/, 2014. Accessed: 2015-02-25.

[Rod15] Richard Rodger. seneca. https://github.com/rjrodger/seneca, 2015.

[Sha10] Remy Sharp. Detecting global variable leaks. https://remysharp.com/2010/10/08/what-is-a-polyfill, October 2010.

[slia] Slides. https://slides.com/.

[slib] sli.do. https://www.sli.do/.

[soc] Socrative. https://www.socrative.com/.

[Szo11] Bartek Szopka. impress.js. https://github.com/impress/impress.js, 2011.

[Wil18] Aleksander Williams. Dust.js. http://www.dustjs.com/, 2018.

[WODW15] Hannah Wolfe, John O'Nolan, Paul Davis, and Jason Williams. Ghost. https://ghost.org/, 2015.

[WW18] w Web Workers. Web Workers. https://html.spec.whatwg.org/multipage/workers.html/, 2018.

[Zak16] Alon Zakai. sql.js-SQLite compiled to javascript. https://github.com/kripken/sql.js/, 2016.

[ZK18] Piotr Zalewa and Oskar Krawczyk. JSFiddle. https://jsfiddle.net/, 2018.

Student Projects

[236] Patrick Balestra, Alexander Fischer, Grigoriy Rebinskiy, Kyriakos Stylianopoulos, and Marco Tollini. asq-java-q. *Web Atelier III Student Project*, 2016.

[237] Valerie Burgener. ASQ progress bar and ASQ canvas. *UROP project*, 2017.

[238] Jacques Dafflon. ASQ peer assessment. *BSc thesis*, 2014.

[239] Gustavo Graziani. A diagram editor question type for ASQ. *BSc thesis*, 2018.

[240] Georgios Kokosioulis. ASQ editor - wysiwyg editor for impress.js presentations. *Erasmus placement project*, 2014.

[241] Daniele Lo Preiato. ASQ.it - instant questions for the ASQ web application. *BSc thesis*, 2018.

[242] Marco Ravazzini. ASQ editor. *BSc thesis,* 2016.

[243] Max Von Bülow. ASQ mobile. *BSc thesis,* 2013.

[244] Nie Zhenfei. Web components in ASQ. *MSc thesis*, 2015.

Acronyms

ACL Access Control List. 49, 178

AJAX Asynchronous JavaScript And XML. 30

ARS Audience Response Systems. 12, 27, 28

BPMN Business Process Model and Notation. 172

CSS Cascading Style Sheets. 30, 36–38, 43, 51, 53, 59, 127, 150, 160, 164, 166

DDD Domain Driven Design. 176

EEG Electroencephalography. 118

ER Entity–relationship. 172

GUI Graphical User Interface. 120, 125, 129, 146

HTML Hypertext Markup Language. 35–38, 40, 51, 53, 55, 61, 71, 117, 127, 128, 131, 132, 147, 150, 160, 166, 175, 176, 179, 180, 183, 184

HTTP Hypertext Transfer Protocol. 30, 71, 160, 176, 177, 179

I/O Input/Ouput. 97

JSON JavaScript Object Notation. 137, 146, 166, 169

LSM Learning Management Systems. 121

MOOC massive open online course. 17

MOOCs massive open online courses. 1, 119, 121

NASA-TLX NASA Task Load Index. 65, 77, 78

OS Operating System. 97

PBL Problem-based learning. 11, 123

PDF Portable Document Format. 179, 180

PNG Portable Network Graphics. 135

QR Quick Response. 133

RQ Research Question. 3, 81, 114, 115

STEM Science, Technology, Engineering and Mathematics. 4, 14, 15, 48, 123

SUS System Usability Scale. 65, 75–78

UID Unique Identifier. 176, 179–181

UML Unified Modeling Language. 172

URL Uniform Resource Locator. 6, 117, 121, 125, 128–130, 133, 156, 179, 180