Imperial College London

BEng Individual Project - Final Report

jSCAPE - Java Self-assessment Center of Adaptive Programming Exercises

Author:
Alexis Chantreau

Supervisor:
Dr. Tristan Allwood

June 12, 2014

Abstract

Abstract here...

Acknowledgements

Thanks to...

Contents

List of Figures

List of Code Listings

Chapter 1

Introduction

1.1 Motivation

- The rise in MOOCs such as Coursera, Udacity shows that there is a real interest in learning programming.
- Programming is considered "difficult" to learn and it can only be learnt effectively through lots and lots of practice.
- This is even more apparent when students are introduced to other programming languages with different paradigms such as Haskell and Prolog, or more low level programming languages such as C.
- Hence, the need to provide a platform for students to practice their programming skills and understanding of programming concepts.
- Having a lecturer come up with all the exercises by himself can be both time consuming and ineffective: some exercises may not be challenging enough for certain top students, or on the contrary too difficult for struggling students, which can be discouraging.
- Lecturers can only rely on a few homework assignments to get an idea of how students are doing. Having a way for lecturers to gather large amounts of data about students' performances would be beneficial. Allows for supplementary material, exercises, etc...

1.2 Objectives

Having identified the problems associated with teaching and learning programming, we were lead to formulating objectives in order to make the

project successful and useful to the parties involved.

The main objective of the project was to produce a web-based teaching infrastructure to complement the introductory first year programming classes. Four key features were identified:

- **Programming questions/exercises -** The web platform should allow students to practice their programming skills and understanding of programming concepts. There should be no limit to the number of questions a student can answer, so that if a student desires more practice, then he should be able to do that. Additionally, it should be possible for a specific set of people, such as lecturers and tutors, to add questions/exercises to the system.
- **Progress tracking -** Designated people, such as lecturers and tutors, should have access to detailed statistics about the students performances. This will provide them with useful information about difficulties particular students, or the entire class, may be facing. In addition, the system should give feedback to the students, in the form of simple statistics, allowing them to identify their weak areas and thus improve on them.
- **Adaptive difficulty -** The questions or exercises presented to the students should be suited to their ability. Not only will this stimulate the learning process, but it will also give a better indication of a student's understanding of the programming concepts being tested.
- **Automated generation -** There should be a mechanism to allow for some degree of automated or semi-automated generation of exercises. This will provide a large supply of "fresh" questions, so that students don't end up answering the same questions and memorizing the answers to them.

While investigating existing solutions (Section 2.4 - Related Work) we found out that some of these features were less common than others. The availability of programming exercises and progress tracking are very essential in such systems, therefore many of the related software we looked at implemented these features. On the other hand, relatively few tools integrated some form of adapting the questions to the students' ability. Finally, almost none of the tools featured automated generation of questions, opting instead to allow exercises to be added manually to the system.

1.3 Contributions

Within the context given above, this project makes the following contributions:

1.4 Report Structure

Chapter 2

Background

2.1 Computer Based Tests

CBT abbreviation - offers advantages such as being able to display higher quality visuals such as pictures, videos, graphs, etc... - low paperwork, everything is stored on the computer - automatic grading, less work for teachers - generation of statistics is made easier by the fact that the data can be processed by the computer - nowadays a lot of learning is done on computer systems, and children are used to dealing with computers so assessment through this medium is advantageous.

CBTs are typically "fixed-item" tests where all the students answer the same set of questions, usually provided by the person responsible for the assessment. This isn't ideal since students can be presented with questions that are too easy or too difficult for them to answer. Consequently, the results of the test won't give a very accurate representation of a student's ability, and for this reason, these types of tests aren't extremely useful. This problem lead to research and the development of computerized adaptive testing (CAT).

2.2 Computerized Adaptive Testing

Computerized adaptive testing (CAT), also called *tailored testing*, is a form of computer-based testing which administers questions (referred to as *items* in the psychometrics literature) of the appropriate difficulty by adapting to the examinee's ability. For example, if an examinee answers an item correctly, then the next item presented will higher on the difficulty scale. On the other hand, if they answer incorrectly, they will be presented with an item lower on the difficulty scale.

From an architectural perspective, a computerized adaptive test (CAT) consists of five components [3]:

1. Calibrated item pool

An item pool is needed to store all the items available for inclusion in a test. This item pool must be calibrated with a psychometric model. During this phase, the item parameters are estimated according to the chosen model and scaled to fit with already existing items. Usually, the psychometric model employed in these systems is called Item Response Theory (IRT) (section 2.3.4). Calibration is a complex process, and to be done accurately it requires a considerable amount of data. Typically, it is performed by psychometricians, aided by expensive and sophisticated calibration software.

2. Starting point

Initially, when zero items have been administered, no information is known about the examinees and so the CAT is unable to estimate their ability. As a result, the item selection algorithm will fail to choose the next item to be administered. If there is previous information available, for example an examinee's ability estimate in a closely related subject, then this can be input into the system to form the starting point configuration. Often this data isn't available or too costly to collect, so the CAT's initial ability estimate for the examinee corresponds to the mean on the ability scale - hence the first item presented will be of average difficulty.

3. Item selection algorithm

The item selection algorithm chooses the next item to present to the examinee based on the ability estimate of the examinee up to that point. Several methods exist and largely depend on the psychometric model in use. One of the most commonly used methods is the *maximum information method* (section 2.3.4), which selects the item which maximizes the information function with respect to the estimated ability at that point.

4. Scoring algorithm

The scoring algorithm refers to the steps taken to update the examinee's ability estimate after an item has been answered. The two most commonly used methods are *maximum likelihood estimation* (section 2.3.4) and *Bayesian estimation* (section 2.3.4).

5. Termination criterion

The termination criterion specifies when the CAT should finish. For example the CAT can terminate when the change in the ability estimate after each iteration is below a certain threshold, or when time has run out, or when N items have been administered, etc... Obviously, the CAT shouldn't be terminated too early, so as to allow enough time to estimate the examinee's ability with acceptable accuracy.

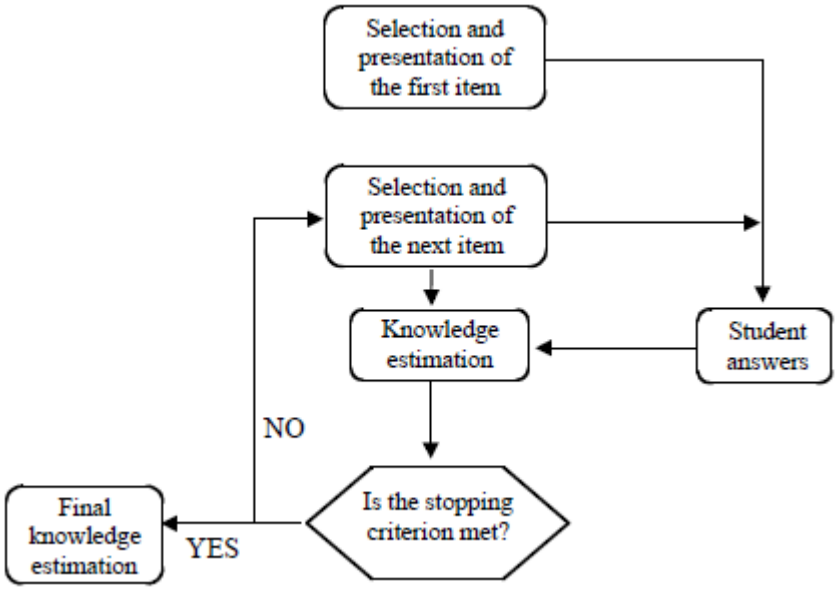


Figure 2.1: Flowchart of an adaptive test. Adapted from [1].

The flowchart in figure 2.1 corresponds to components 2-5, and illustrates the basics of the algorithm implemented in CAT. [2] gives a more detailed description of the procedure:

1. The pool of items that haven't been administered yet is searched to determine the best item to present to the examinee, according to the current estimation of his ability.

2. The chosen item is presented to the examinee, who then answers it correctly or incorrectly.

3. The ability estimate is updated, based upon this new piece of information and the previous ability estimate.

4. Steps 1–3 are repeated until a termination criterion is met.

5. The algorithm returns a final ability estimate for the examinee's performance along with a confidence level: a percentage value indicating how accurate the estimate is.

CATs offer several advantages over traditional CBTs. As a result CATs have been used in many areas[4], such as education, job hiring, counselling, clinical studies, etc... Since CATs administer items by adapting to the examinee's ability, the test-taking experience ends up being a more positive one. Indeed, examinees won't have to deal with answering items which are too difficult or too easy compared to their ability level, a problem which appears in traditional CBTs.

In addition, by administering only those items which will yield additional information, CATs end up being more accurate in estimating an examinee's ability level. This contrasts with CBTs which usually provide the best precision for examinees of medium ability, whereas extreme scores end up being less accurate.

Lastly, CATs can come up with an ability estimate much quicker and with fewer administered items when compared to traditional CBTs. Indeed, an adaptive test can typically be shortened by 50% and still maintain a higher level of precision than a fixed version.[5]

Despite the advantages mentioned above, CATs have some limitations. A frequent complaint is that an examinee isn't allowed to go back and change his answer to a past item. This limitation exists to prevent the examinee from intentionally answering items incorrectly to make subsequent items easier, and then going back and selecting the correct answers to achieve a perfect score. For similar reasons, it isn't possible to skip items, the examinee must select an answer to move on to the next item.

The second issue has to do with the items themselves. First of all, there is the need for a large bank of items to cater to all ability levels. Developing an item pool of sufficient size can be very time consuming. David J. Weiss writes in [6] that item pools with 150-200 items are to be preferred, although 100 high quality items can sometimes be enough to achieve adequate estimations of ability levels.

Secondly, for the CAT to be of good quality the item pool needs to be calibrated accurately. This requires pre-administering the items to a sizeable sample and then simultaneously estimating all the item parameters for each

item. The guidelines in [7] suggest that sample sizes may be as large as 1000 examinees. This phase is costly, time consuming and often times simply unfeasible.

Lastly, item exposure is a possible security concern. Sometimes particular items may be presented too often and become overused. This may result in examinees becoming familiar with them and sharing them to other examinees of the same ability level, thus corrupting the results of the test. This problem can be solved to some extent by modifying the item selection algorithm to include some exposure control mechanism.

A brief overview of CATs was given in this section. All of these concepts will be explored in more detail in item response theory (section 2.3.4) and in the implementation of adaptive testing in jSCAPE (section 5.??).

2.3 Probabilistic Test Theory

In the previous section we listed some of the components necessary for the development of CATs. Many of these components, especially the item selection and scoring algorithms, rely heavily on probabilistic concepts. Therefore, in this section we go over a few topics in probability and how they can be implemented in a psychometric model to be used in computerized adaptive testing.

2.3.1 Probability Theory

Probability theory provides us with a means to model uncertainty in data and to infer information from observed data. This is especially useful when it comes to estimating latent variables, i.e. variables that are not directly observed, instead they are inferred from other observed variables. For instance, examinee ability is a latent variable, hence the reason for section 2.3.

The probability of an event E occurring is a numerical value between 0 and 1, indicating how probable it is that we will observe event E. It is denoted as $P(E)$ and $0 < P(E) < 1$. The value 1 indicates total certainty, whereas the value 0 indicates impossibility.

In addition, there is the concept of conditional probability where the probability of an event E_1 occurring, given that event E_2 has occurred, can be denoted as $P(E_1|E_2)$. This concept is also important in statistical inference

because it allows us to update prior beliefs given additional observed data. This is explained in more detail in sections 2.3.2 and 2.3.3.

2.3.2 Likelihood and Maximum Likelihood Estimation

Although the terms probability and likelihood are used interchangeably in every day life, in statistics a distinction can be made.

For any stochastic process, let us denote the observed outcomes as x and the set of parameters as θ. When we say probability, we want to calculate $P(x|\theta)$, i.e. the probability of observing the outcomes x given specific values for the set of parameters θ.

However, sometimes we do not know the specific values for θ, instead, we have observed some outcomes x, and want to find out how likely a particular value of θ is given the observed outcomes x. We call this the likelihood or likelihood function, and it is denoted as $L(\theta|x)$. The likelihood of a set of parameter values, θ, given outcomes x, is equal to the probability of those observed outcomes given those parameter values, i.e. $L(\theta|x) = P(x|\theta)$.

To highlight the distinction we illustrate with an example of how the two terms are used. If we consider a dice, a possible parameter is the fairness of the dice, while possible outcomes are which values are displayed after a roll. For instance, if a fair dice is rolled 5 times, what is the *probability* that a 6 will show up on every roll? If a dice is rolled 5 times and lands on 6 every roll, what is the *likelihood* that the dice is fair?

Maximum likelihood estimation refers to a method of statistical inference where one can find the set of parameter values (θ) which are most *likely* given the observed outcomes (x). As mentioned in the name of the method, this is done by finding the parameter values which maximize the likelihood function $L(\theta|x)$.

2.3.3 Bayesian inference

Bayesian inference is a method of statistical inference where Bayes' theorem is used to update the probability estimate for a hypothesis after observing additional events or outcomes.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Let us consider a small example: There is a 60% chance of it snowing on Thursday. If it snows on Thursday, there is a 20% chance of it snowing on Friday. If it didn't snow on Thursday, there is a 70% chance it will snow on Friday.

Before observing any additional data, we can say that the probability of it snowing on Thursday is 60%. But, we are given additional information, namely that it snowed on Friday. How we can we update the probability that it snowed on Thursday to reflect this observation?

We can do this by using Bayes' theorem shown in equation (2.1). Let A be the event "snowing on Thursday" and B be the event "snowing on Friday". Then we have:

$$P(A|B) = \frac{0.2 \times 0.6}{0.2 \times 0.6 + 0.7 \times 0.4} = 0.3$$

This example shows how additional observed data can affect one's prior beliefs. The observation that it snowed on Friday reduced the probability that it snowed on Thursday from 60% to 30%.

However, in the context of this project, we are interested in Bayesian inference of the probability distributions of parameters.

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

The prior distribution, $P(\theta)$, is the distribution of the parameters θ before any data is observed. $P(x|\theta)$ is the distribution of the observed data conditioned on the parameters. The posterior distribution, $P(\theta|x)$, is the probability distribution of the parameters after considering the new information brought by the observed data.

2.3.4 Item Response Theory

We mentioned in section 2.2 that Item Response Theory (IRT) is usually the psychometric model of choice when developing a CAT, e.g. the Graduate Record Examination (GRE) and Graduate Management Admission Test (GMAT). CATs can still be implemented with Classical Test Theory but they offer less sophistication and less information to evaluate/improve the reliability of the test, making IRT the superior choice.

In psychometrics, an item is a generic term used to refer to a question or an exercise. For instance, in a mathematics exam, "What is the square root of 81?" is a possible item.

Item Response Theory gets its name from focusing on analyzing the items themselves as opposed to Classical Test Theory, which considers the test as a whole. Indeed, Classical Test Theory judges an examinee's ability simply on the number of correct answers obtained, totally disregarding which items were answered correctly or incorrectly, thereby making the assumption that all items possess identical properties.

IRT hinges on the idea that it is possible to model, as a mathematical function, the probability of a certain response to an item given the item parameters and the examinee's ability level. IRT is based on a set of strong assumptions[8]:

1. A unidimensional trait denoted by θ;
2. Local independence of items;
3. The response of a person to an item can be modelled by a mathematical *item response function* (IRF).

The unidimensionality of the trait means that the items are supposed to measure one characteristic of the person, generally their ability, and that this trait should account for most of the variance in the test score. The trait level is denoted as the Greek letter theta (θ), and typically it has a mean of 0 and a standard deviation of 1. For example, in the context of this project, theta (θ) will represent an examinee's ability level in a particular programming subject (e.g. arrays, binary trees, etc...). Ability is a latent variable (section 2.3.1) because it isn't directly observable, therefore it must be inferred from the observable, concrete data such as examinee responses.

The local independence of items states that an examinee's responses to items are independent of one another. This property of conditional independence is crucial to estimating examinee trait levels as we will see later on in this section.

Several IRT models have been developed over the years to address the different types of tests that exist, e.g. multiple choice exams, agreement questionnaires (Likert scale), etc... These models all seek to achieve the same goal, modelling the examinee's ability on some ability scale, but they differ

in the number of parameters associated with each item. We shall note that we only consider unidimensional IRT models which deal with dichotomously scored items, i.e. scored as correct or incorrect, in a multiple choice question for instance. We now take a look at these different models in more detail.

The one-parameter logistic model

Every IRT model is defined by two elements: a set of item parameters, and a mathematical item response function. This function plots the probability that an examinee of a given ability will answer a particular item correctly. The one-parameter logistic (1PL) model is the simplest IRT model because in this model items are only characterized by one parameter: the difficulty parameter b_i, where the subscript i identifies item i. This has the effect of making the item response function quite simple.

$$P_i(\theta) = \frac{1}{1 + \mathrm{e}^{-(\theta - b_i)}} \tag{2.2}$$

Equation (2.2) shows the item response function for the 1PL model. θ is the examinee's ability level, $P_i(\theta)$ is the probability of answering item i at all ability levels, and as mentioned above, b_i is the item difficulty.

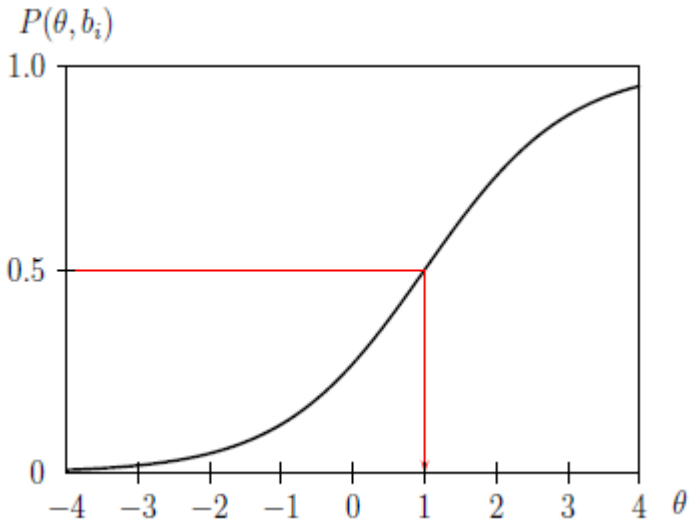


Figure 2.2: The item response function of the 1PL model. (Source:[10]).

Figure 2.2 shows the probability of answering an item correctly for all ability levels. In the 1PL model, the item difficulty is the point at which a correct response and an incorrect response are equally probable. In the figure this is

shown in red, and in this case, the item has difficulty parameter $b_i = 1$. This highlights one of IRT's attractive properties, the fact that examinee ability and item difficulty are both measured on the same scale. The ability scale is a design choice, in this case it ranges from -4 to +4.

Ability (θ)	-4	-3	-2	-1	0	1	2	3	4
$P_1(\theta)$	0.007	0.018	0.047	0.119	0.269	0.5	0.731	0.881	0.953
$P_2(\theta)$	0.047	0.119	0.269	0.5	0.731	0.881	0.953	0.982	0.993

Table 2.1: Probabilities of a correct answer for two items in the 1PL model.

Table 2.1 shows the probability values for two 1PL items. $P_1(\theta)$ corresponds to the probability of answering item 1 (the item in figure 2.2) correctly. $P_2(\theta)$ corresponds to the probability of answering item 2 (with difficulty $b_2 = -1$) correctly. It can be seen that as the ability level of the examinee increases, so does the probability of him answering the item correctly. The difficulty parameter shifts the item response function to the left or to the right depending on its value.

Although the 1PL model provides a good basis for IRT it is rather simplistic and fairly limited. For instance, the model assumes that at low ability levels the examinee has close to no chance to answer the item correctly. However, in multiple choice questions there is always the option of guessing randomly. These existing limitations bring us to considering the next model.

The two-parameter logistic model

The two-parameter logistic (2PL) model builds upon the 1PL model by adding a new item parameter: the discrimination (a_i), where the subscript i identifies item i. Discrimination refers to an item's capacity to distinguish between examinees of different ability levels. Graphically, the discrimination parameter corresponds to the slope of the item response function, and typically it ranges from -2.8 to 2.8.

$$P_i(\theta) = \frac{1}{1 + \mathrm{e}^{-1.7a_i(\theta - b_i)}} \tag{2.3}$$

Equation (2.3) shows the item response function for the 2PL model. It is very similar to the equation of the 1PL model (Equation (2.2)), except for the inclusion of the discrimination parameter (a_i) and a constant of 1.7 to control the shape of the curve.

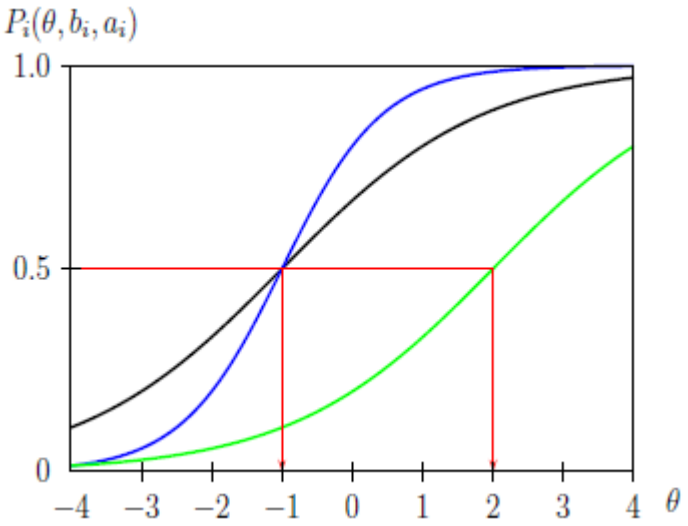


Figure 2.3: The item response functions of three items in the 2PL model. (Source:[10]).

Figure 2.3 shows the item response functions of three items in the 2PL model. As in the 1PL model, the item difficulty is still located where $P_i(\theta) = 0.5$, shown in the red lines. The item represented by the black curve and the item represented by the green curve both have the same discrimination, but different difficulty. As mentioned before, all this does is shift the item response function to the right or to the left, depending on which item you take as the reference point.

The black and blue curves illustrate the other scenario. The item represented by the black curve and the item represented by the blue curve have different discrimination, but identical difficulty. This has the effect of making the blue curve much steeper than the black one, thus affecting the probabilities over the ability scale. Indeed, the item represented by the blue curve discriminates quite well between respondents. Examinees of low ability have a much lower probability of answering the item correctly than examinees of higher ability. Compare this to the item represented by the black curve, where examinees of low ability still have a fair chance of getting the item correct.

The three-parameter logistic model

The three-parameter logistic (3PL) model takes into account the possibility of guessing a correct response to an item. This model is especially convenient for tests where multiple choice questions are present. In IRT, this parameter

is called the pseudo-guessing or chance parameter, and it is denoted as c_i, where the subscript i identifies item i. Graphically, it corresponds to a lower asymptote, i.e. $P_i(-\infty) = c_i$. The guessing parameter does not vary as a function of the examinee's ability. Indeed, whether they be of low ability or high ability, examinees both have the same probability of guessing the correct answer to an item.

$$P_i(\theta) = c_i + (1 - c_i)\frac{1}{1 + \mathrm{e}^{-1.7a_i(\theta - b_i)}} \tag{2.4}$$

Equation (2.4) shows the item response function for the three-parameter logistic model (3PL). The addition of c_i defines a lower asymptote at that value, whereas $1 - c_i$ acts as a weighting factor towards the 2PL model equation (Equation (2.3)). Typically the pseudo-chance parameter ranges from 0 to 0.35, greater values being considered unacceptable. Lastly, if $c_i = 0$ then we are simply left with a normal 2PL model.

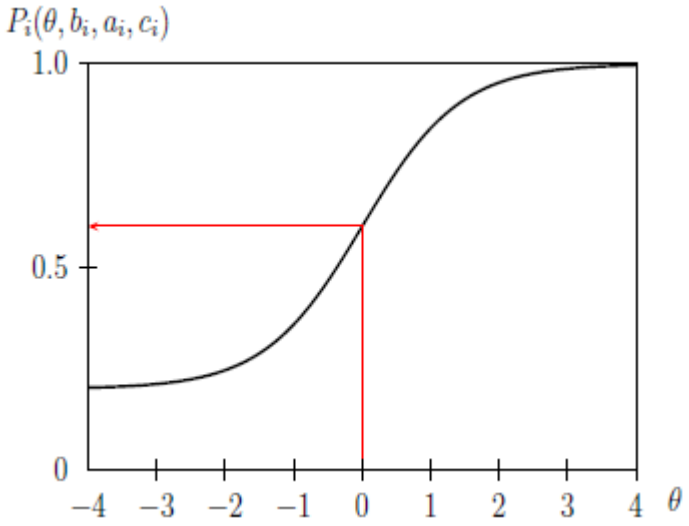


Figure 2.4: The item response function of the 3PL model. (Source:[10]).

Figure 2.4 shows the item response function of an item in the 3PL model. This item has parameters $a = 1.4$, $b = 0$ and $c = 0.2$. Now it can be seen that the guessing phenomenon is taken into account by this IRT model. Indeed, at the lowest possible ability ($\theta = -4$), the probability of a correct answer is $P(-4) = 0.200059$, which is very close to the guessing parameter of the item.

Something to note is that $P_i(b_i) \neq 0.5$, i.e. the probability of a correct response at the ability level equal to the item difficulty is no longer 0.5.

Instead, we have $P_i(b_i) = c + \frac{1-c}{2}$, and in figure 2.4 we have $P(0) = 0.6$. If we consider a multiple choice question, adapted to the examinee's ability level, then it makes sense that the probability of them getting it correct would be greater than 0.5, to account for the possibility of getting it right merely through guessing.

Item information

In section 2.2, when discussing item selection algorithms, we mentioned the maximum information method as a possible implementation. This method relies on the IRT concept of item information.

In psychometrics and statistics, the term *information* is defined as the reciprocal of the precision with which a parameter can be estimated[9]. Therefore, item information is the precision in the ability estimate that the item provides, at all ability levels. It is also an indication of the quality of the item in terms of how well that item can discriminate between several respondents.

The item information function for the 1PL model is calculated as

$$I(\theta) = P_i(\theta)Q_i(\theta),$$

where $Q_i(\theta) = 1 - P_i(\theta)$.

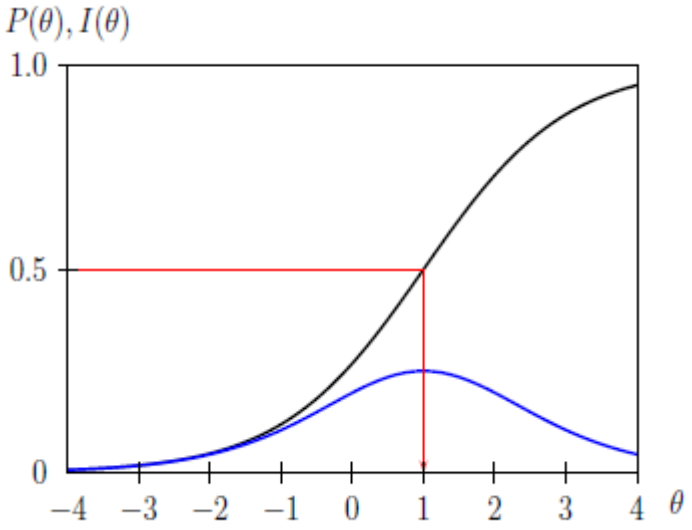


Figure 2.5: Item response function and item information function for the 1PL model. (Source:[10]).

In figure 2.5, two curves are plotted. The black curve is the item response function of an item, and the blue curve is the associated information function for that item. In the 1PL model, the maximum value of item information occurs where the probability of a correct answer and the probability of an incorrect answer are both equal to 0.5. This point is indicated by the red line. If we recall earlier, this point also represents the difficulty of the item, i.e. item parameter b_i. Thus, the item provides the most information for those examinees whose ability is equal to the difficulty of the item, 1 in this case. This means that administering this item will give more precise ability estimates for examinees whose true ability is at that particular level. Presenting this item to examinees not at that particular level, examinees of ability -2 for instance, will not yield very much precision to subsequent ability estimates.

The item information function for the 2PL model is calculated as

$$I(\theta) = a_i^2 P_i(\theta) Q_i(\theta),$$

where $Q_i(\theta) = 1 - P_i(\theta)$. In the 2PL and 3PL models, the discrimination parameter, a_i, plays a significant role in the function, as it appears squared in the function.

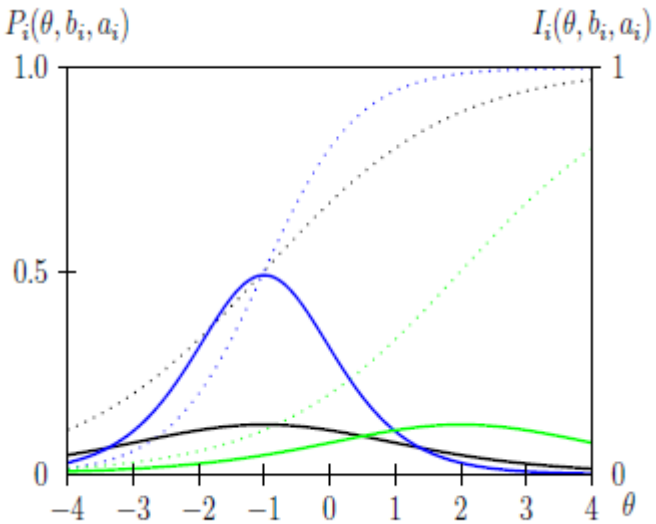


Figure 2.6: Item response functions and item information functions for three items in the 2PL model. (Source:[10]).

Figure 2.6 shows three 2PL item response functions, in dotted lines, matched in color with the corresponding item information functions in solid lines. Like

in the 1PL model, items still reach their maximum information at the point where ability is equal to the difficulty of the item. However, the amount of information provided at that ability level can now vary depending on how large the discrimination parameter is.

The items represented by the black and blue curves both have the same difficulty parameter. But, the item represented by the blue curve has a higher discrimination parameter, and therefore this affects the item information function. Higher discrimination values lead to more item information, whereas lower discrimination values make the item less informative.

The items represented by the black and green curves both have the same discrimination parameter, so the maximum information value will be the same for these items. However, these items do not share the same difficulty parameter. All this does is shift the curve along the ability axis, thus changing the location of maximum information for that item.

The item information function for the 3PL model is calculated as

$$I(\theta) = a_i^2 \cdot \frac{(P_i(\theta) - c_i)^2}{(1 - c_i)^2} \cdot \frac{Q_i(\theta)}{P_i(\theta)},$$

where $Q_i(\theta) = 1 - P_i(\theta)$.

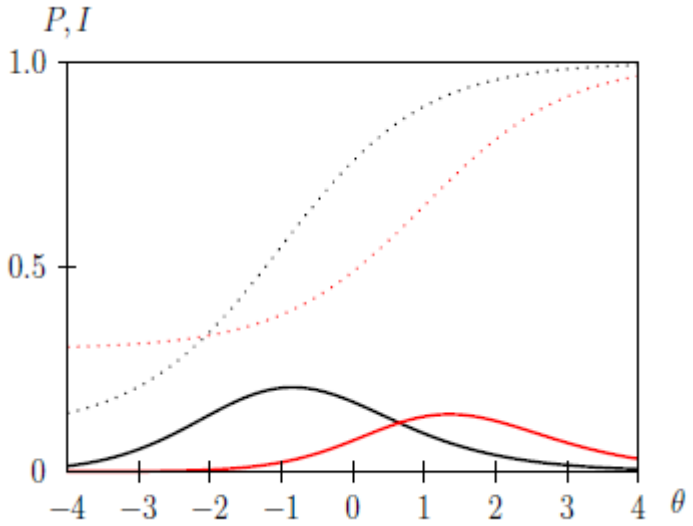


Figure 2.7: Item response functions and item information functions for two items in the 3PL model. (Source:[10]).

Figure 2.7 shows two 3PL item response functions, in dotted lines, matched in color with the corresponding item information functions in solid lines. In the 3PL model, the maximum of item information functions is no longer at the point where ability is equal to the difficulty of the item. Here, the guessing or pseudo-chance parameter (c_i) plays a role in the shape of the item information function. Higher guessing parameters lead to less item information, whereas lower guessing parameters lead to more item information. For instance, the item represented by the black curve has $c_i = 0.1$, and the item represented by the red curve has $c_i = 0.3$, which explains why the item represented by the black curve yields more information.

Estimating the ability

In section 2.2, when discussing scoring algorithms, we saw that the two most common methods for ability estimation were *maximum likelihood estimation* and *Bayesian estimation*.

In the *maximum likelihood estimation* method, we need to define a likelihood function in terms of the ability level (θ) we are trying to estimate:

$$L(\theta|\mathbf{u}) = L(\theta|u_1, ..., u_n) = \prod_{i=1}^{n} P_i(\theta)^{u_i}(1 - P_i(\theta))^{(1-u_i)},$$

where $\mathbf{u} = (u_1, ..., u_n)$ is called the response vector for an examinee, that is $u_i = 1$ if the examinee answers the i$^{\text{th}}$ item correctly, and $u_i = 0$ if the examinee answers the i$^{\text{th}}$ item incorrectly. $P_i(\theta)$ corresponds to the probability of answering the i$^{\text{th}}$ item correctly when the ability level is θ, and thus $1 - P_i(\theta)$ gives the probability of answering the i$^{\text{th}}$ item incorrectly when the ability level is θ.

Now that the likelihood function is defined, we can apply maximum likelihood estimation to find the ability level which is most likely given the examinee's response vector. Let us illustrate with a very simplistic example where ability levels are discrete values between -1 and $+1$. We would iterate through these ability levels and compute the following values, for example:

$$L(\theta = -1|\mathbf{u}) = 0.001$$
$$L(\theta = +0|\mathbf{u}) = 0.017$$
$$L(\theta = +1|\mathbf{u}) = 0.058$$

The likelihood function is maximized when $\theta = +1$, and so this is the maximum likelihood estimate for θ. In a CAT, and based on the examinee's

response vector $\mathbf{u}$, the system would assign $+1$ as the ability level for that examinee.

In the *Bayesian estimation* method, the CAT maintains a *knowledge distribution* $P(\theta)$, which represents the probability that the examinee's ability is θ. In the absence of any additional information, the CAT considers the examinee's initial knowledge distribution to be a uniform distribution.

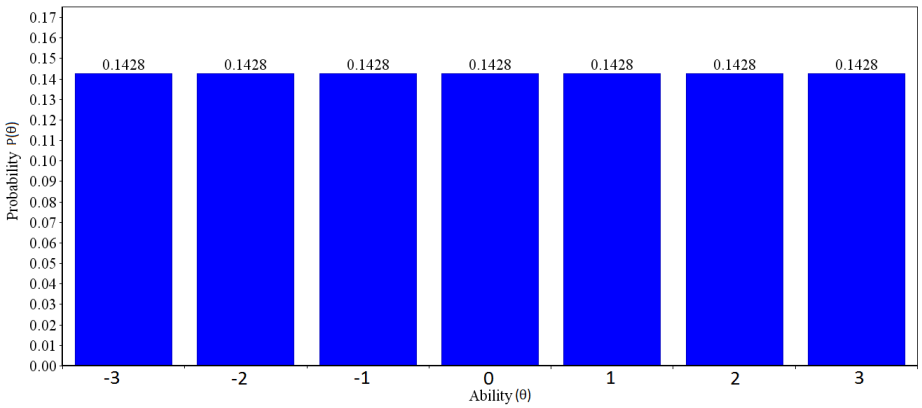


Figure 2.8: Initial examinee knowledge distribution.

For example, figure 2.8 shows the initial examinee knowledge distribution in a CAT having discrete ability levels ranging from -3 to 3. Continuous knowledge distributions are certainly possible as they allow for, in theory, infinitely more ability levels. However, they are more complex and require the use of integration to compute probabilities, thus for the purposes of illustration, we will only consider discrete knowledge distributions.

The *Bayesian estimation* method relies on Bayesian inference to derive the posterior knowledge distribution according to Bayes' rule:

$$P(\theta|\mathbf{u}) = \frac{P(\mathbf{u}|\theta)P(\theta)}{P(\mathbf{u})},$$

where, again, $\mathbf{u}$ is the examinee's response vector and θ corresponds to the possible ability levels.

The posterior distribution is the result of updating the prior knowledge distribution, $P(\theta)$, with observed data, $P(\mathbf{u}|\theta)$, i.e. the examinee's response pattern. We can express this using the likelihood function:

$$P(\theta|\mathbf{u}) \propto L(\theta|\mathbf{u}) \cdot P(\theta)$$

After computing the posterior knowledge distribution, the CAT can estimate the ability estimation for the examinee. This is equal to the mode of the knowledge distribution, i.e. the ability associated with the highest probability value.

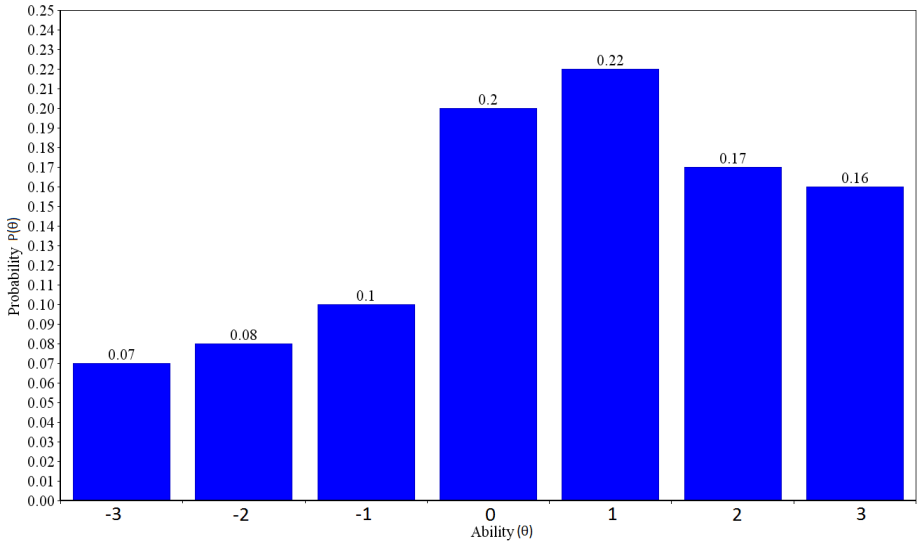


Figure 2.9: Examinee knowledge distribution after answering an item correctly.

For example, figure 2.9 shows the knowledge distribution of an examinee after being shown an item of medium difficulty and answering it correctly. It can be seen that lower ability levels have become less likely, whereas higher ability levels are now more probable.

The ability level associated with the highest probability is $\theta = 1$, thus this becomes the estimated ability. Over time, after the examinee answers more items, the estimation will become associated with even higher probability values. These probabilities indicate how confident the CAT is in that particular ability estimation.

2.4 Summary

In this section, the relevant background literature and theory for this project has been discussed...

We have gone into quite some detail explaining the techniques behind Item Response Theory, and these will become the basis of the implementation of adaptive difficulty in jSCAPE.

We have seen two methods for estimating an examinee's ability level, and several models to represent probabilities of answering items correctly...

We have gone over types of computer based assessment and how they differ from regular assessment techniques. We have also provided an overview of the theoretical components and probability concepts which will be implemented in jSCAPE (chapter xx?).

Chapter 3

Related Work

Web-based/Computer based education and adaptive web-based assessment systems are a "hot" research area, and as a result, numerous tools, environments and infrastructures have emerged. There are common features to all, however some distinguish themselves by having not so common features. Automated exercise generation in these tools is usually non-existent or very limited. Moreover, the tools are more focused on assessing students rather than self-assessment, i.e. students take tests which count towards their final grade on these systems.

There are many components involved in this project, two of the more important ones are adaptive difficulty and automated generation of exercises, so there are many tools which exist which do one or the other, very rarely both.

In this section we look at related software and evaluate them with respect to the objectives listed at the beginning of the development of jSCAPE.

3.1 Environment for Learning to Program

Environment for Learning to Program (ELP) is an interactive web based environment for teaching programming to first year Information Technology students at Queensland University of Technology (QUT).

3.2 CourseMarker

CourseMarker is a re-design of Ceilidh, a computer based assessment system used at the University of Nottingham for 13 years. Ceilidh was quite a

complete system, providing coursework, the management of modules and the presentation of module content.

3.3 Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students

The Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students (AEGIS)

3.4 Programming Adaptive Testing

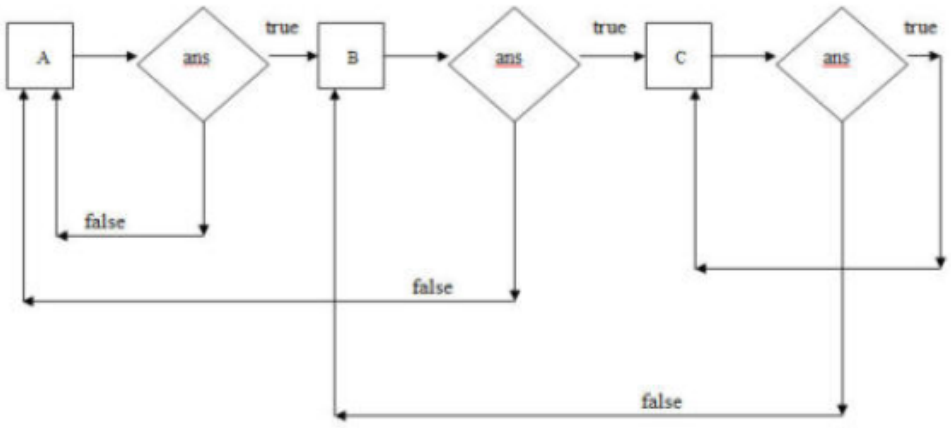


Figure 3.1: Adaptive sequence of questions in PAT. (Source:[11])

3.5 Adaptive Self-Assessment Master

Adaptive Self-Assessment Master (ASAM) is an extension to CourseMarker, which improves upon it by administering questions which are suited to the student's ability.

3.6 System of Intelligent Evaluation Using Tests for Tele-education

The System of Intelligent Evaluation Using Tests for Tele-education (SIETTE) is a web based environment for generating and constructing adaptive tests.

3.7 Summary

We have looked at some of the relevant work in the field of computer based education and assessment. We saw that SIETTE provided many of the features we set out to replicate in jSCAPE, therefore particular parts of our implementation will be inspired by SIETTE.

Chapter 4

The jSCAPE System

The jSCAPE system is designed for two distinct groups of users: students and teachers/lecturers. This separation of roles lead to the development of the main application for students, and an administrator tool for teachers/lecturers.

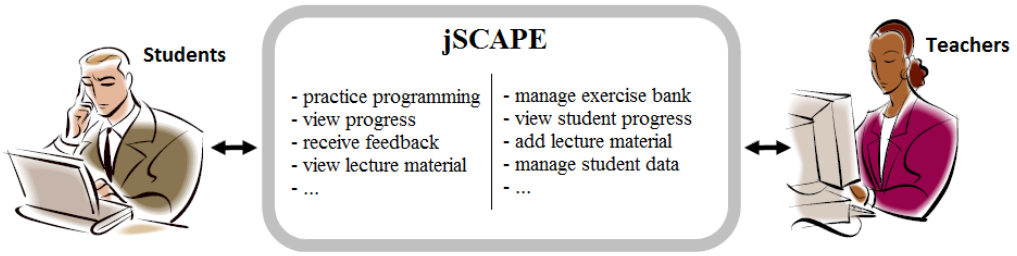


Figure 4.1: Use case diagram of the jSCAPE system.

Figure 4.1 shows some of the main capabilities of the jSCAPE system. Students can practice their understanding of programming concepts by answering exercises provided by the lecturers, and receive feedback while doing so. In addition, students can track their progress by viewing various graphs and charts of their performance on particular exercise categories. Finally, students can access lecture notes and website links provided by the teacher.

Teachers can manage the exercise bank, whether it be adding exercises manually or automatically generating new ones based on templates. They can keep track of their students' progress and thereby identify any difficulties particular students are having. Finally, teachers are responsible for adding lecture material, website links and creating student profiles to store in the database.

In the rest of this chapter we take a closer look at the current available features of jSCAPE.

At the time of writing this report, we would like to note that the screen shots of the application do not represent the final version of jSCAPE, in particular, the graphics and logos haven't been finalized.

4.1 Student view

4.1.1 Login screen

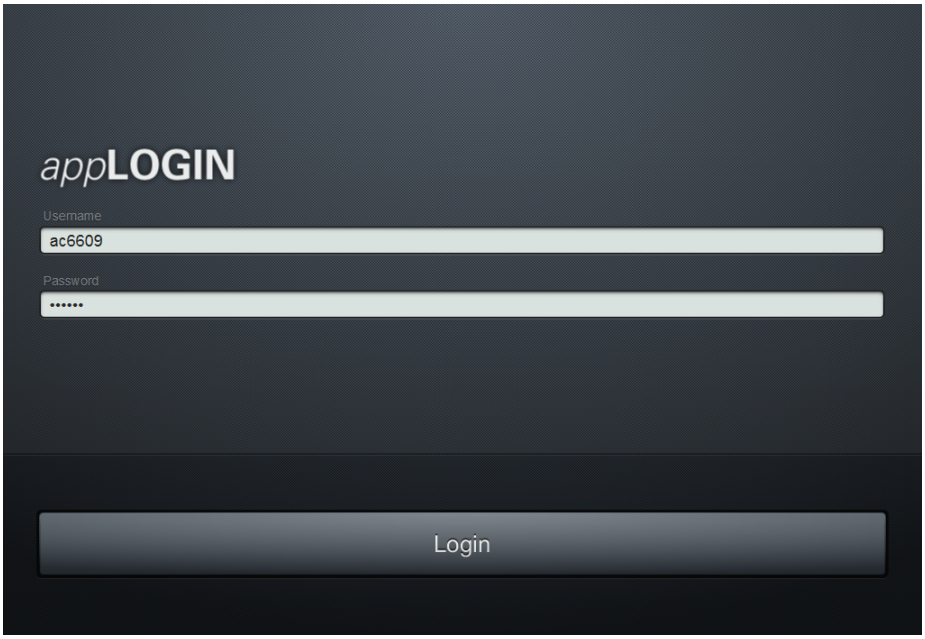


Figure 4.2: The jSCAPE login screen.

For a student to use jSCAPE, they need to be in possession of login credentials, usually acquired by asking the appropriate teacher or lecturer. A connection to the jSCAPE system will be rejected if the entered login details are incorrect. Otherwise, the student can proceed to jSCAPE and access its features.

4.1.2 Tracking progress through statistical data

After a successful login the student lands on the Profile tab which presents information about the student, as well as statistical data on the student's

performance and usage of the system.

Figure 4.3: An overview of the Profile tab in jSCAPE.

Figure 4.3 shows the Profile tab after the student has logged in to the system. Profile information for the student is listed on the left hand side, in the light-blue rectangle. This information includes the student's first name, last name, user name, which class the student is in, the last time the student logged in, and the last time the student answered an exercise.

The main part of the Profile tab is split horizontally between statistical data in the form of pie charts and tables, and graphical data in the form of bar charts.

Figure 4.4 shows the performance of the student in a particular exercise category, in this case "Arrays". In the example, the student has gotten 60% of array exercises correct and thus 40% of them wrong. The student can view the performance pie chart for other exercise categories by changing the selected category in the combo box.

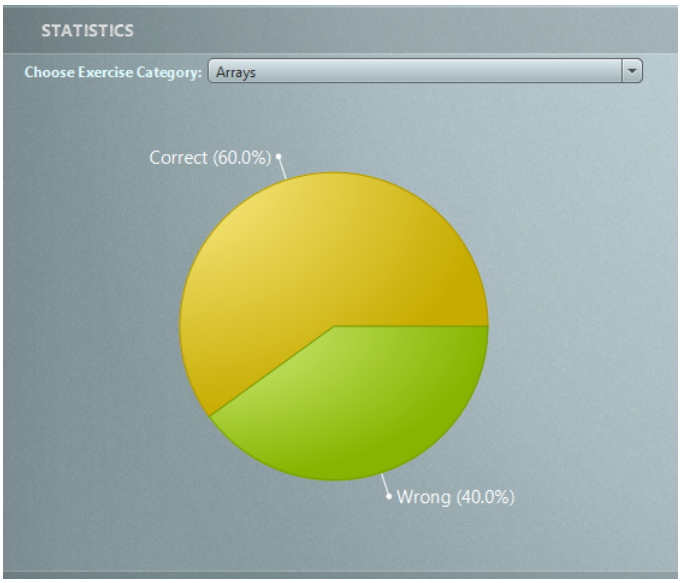


Figure 4.4: Pie chart statistics for exercise category.

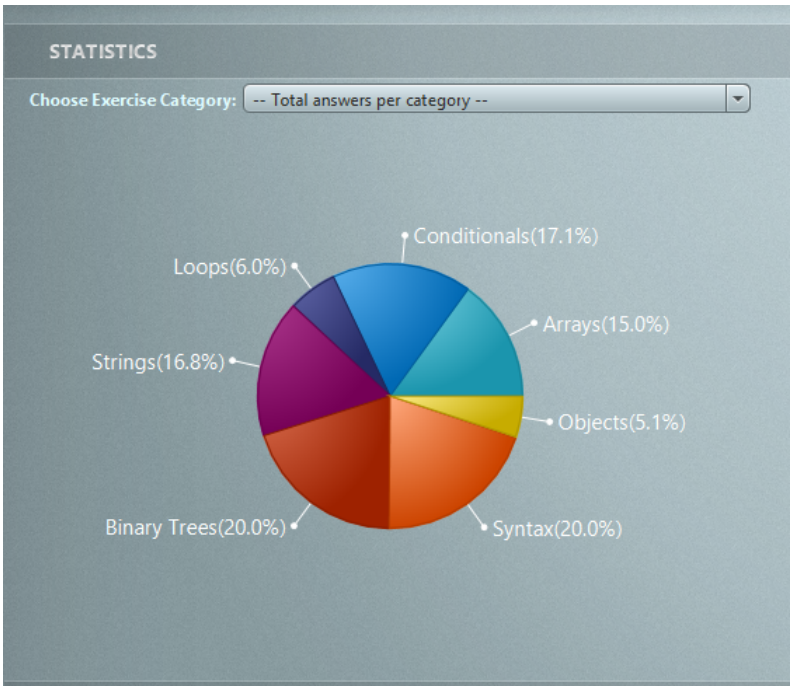


Figure 4.5: Pie chart statistics for distribution of answers.

Another type of pie chart available in jSCAPE can be seen in figure 4.5. This pie chart shows the distribution of answers per exercise category. This is a

useful feature when a student is trying to get a balanced amount of practice in all exercise categories.

Next, performance data is presented to the student in the performance summary table, shown in figure 4.6. In this table, there is a row for every exercise category and a row for the total of each column. Each row contains the number of exercises answered, the number of correct answers and the number of wrong answers associated with a particular exercise category.

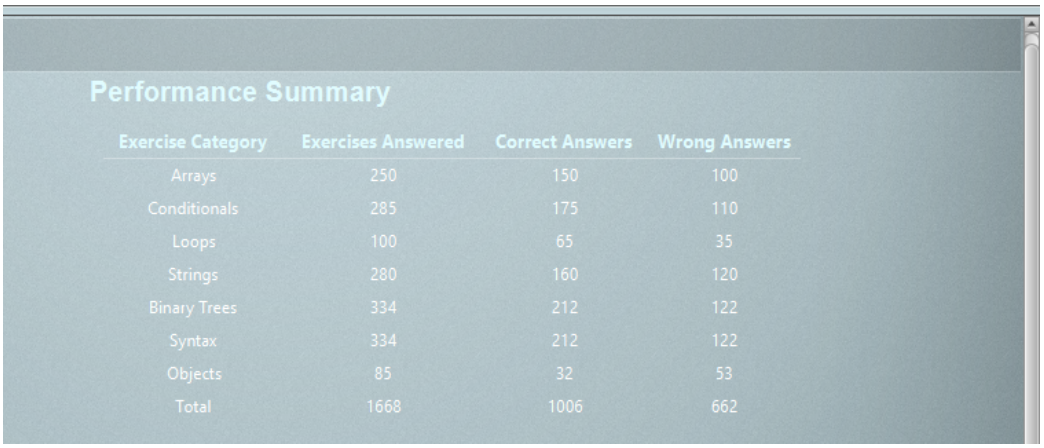


Figure 4.6: Performance summary table.

In the lower half of the Profile tab there is the possibility to view performance data in the form of stacked bar charts.

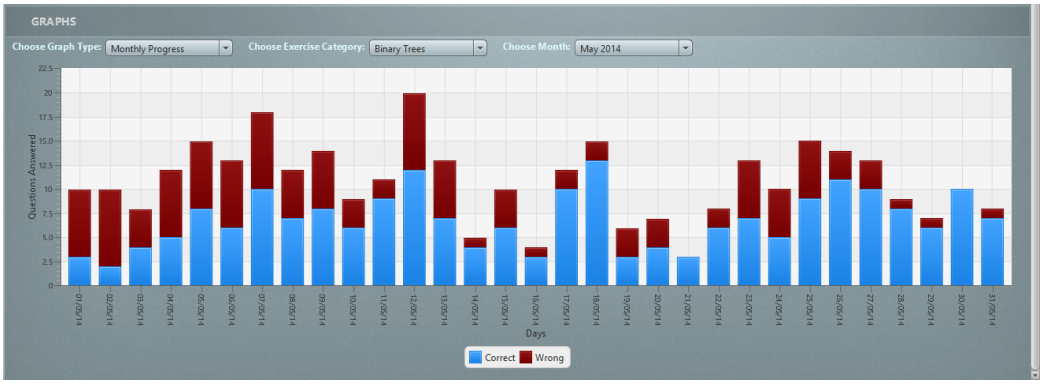


Figure 4.7: Graph data of monthly progress.

Figure 4.7 shows the monthly progress of a student for the month of May 2014 and for the exercise category "Binary Trees". The number of correct

answers (in blue) and wrong answers (in red) are graphed for each day where the student answered exercises. The student can view his monthly progress in other exercise categories and other months by manipulating the appropriate combo boxes. This historical data goes back to the first month in which the student answered an exercise.

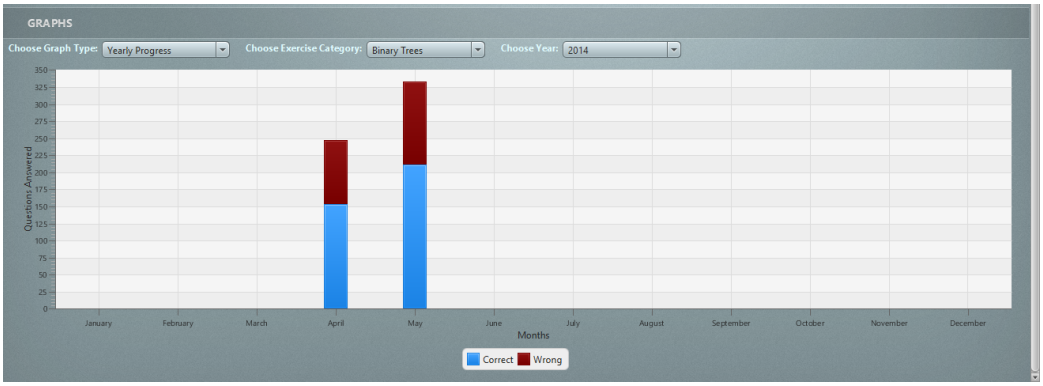


Figure 4.8: Graph data of yearly progress.

Figure 4.8 shows the yearly progress of a student in 2014 for the exercise category "Binary Trees". For each month where the student answered exercises, a stacked bar can be found containing the total number of correct answers (in blue) and the total number of wrong answers (in red) for that particular year and exercise category. The student can view his yearly progress in other exercise categories and other years by manipulating the appropriate combo boxes. This historical data goes back to the year in which the student first started answering exercises.

4.1.3 Practising programming

Selecting the Practice tab brings the student to a window where exercise categories, defined by the teacher, are displayed. These categories exist to separate exercises so that students can focus on practising one particular concept at a time.

Figure 4.9 gives an overview of the Practice tab in jSCAPE. In this case, seven exercise categories have been defined by the teacher: Arrays, Loops, Syntax, Conditionals, Binary Trees, Strings and Objects. Order is irrelevant, students simply choose what type of exercise they want to practice.

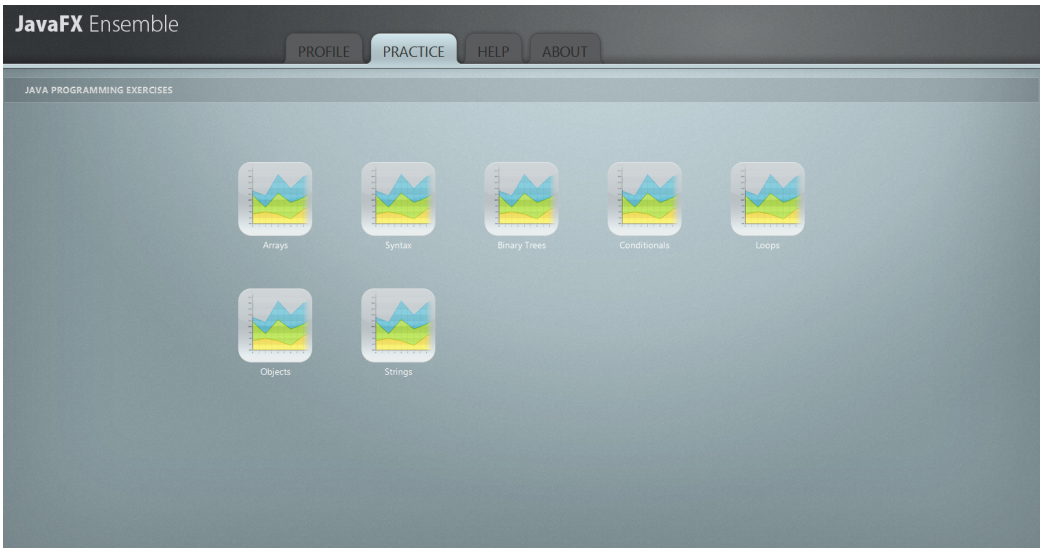


Figure 4.9: An overview of the Practice tab in jSCAPE.

Clicking one of the exercise categories will bring the student to a new window, with an exercise and some helpful information about the chosen exercise category. Figures 4.10 and 4.11 give examples of what this exercise view can look like.

In the case of the binary tree exercise (figure 4.10), exercise data, in the form of a binary tree, is displayed on the left of the window. On the right side of the window is the exercise itself, in this case it asks what should be printed if the binary tree is traversed using the in-order algorithm, and gives four options to choose from.

In the case of the exercise on conditionals (figure 4.11), exercise data, in the form of a code fragment, is displayed on the left of the window. On the right side of the window is the exercise itself, in this case it asks the student to examine the code and to determine the final values of two variables, and provides again four options to choose from.

On the far right of every exercise, there is a dark blue sidebar which displays a description of the programming concept or construct being practised, links to university or course lecture notes and some other websites on the Internet if further help is needed. An example sidebar is shown in figure 4.12, in this case the sidebar for the "Binary Trees" exercise category.

Example exercises

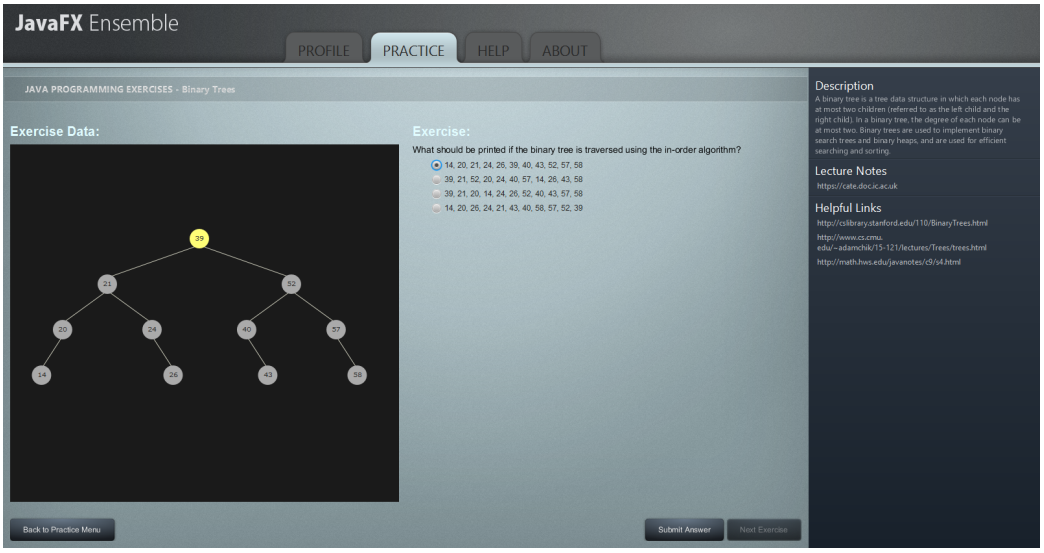


Figure 4.10: The Practice tab view showing an exercise on binary trees.

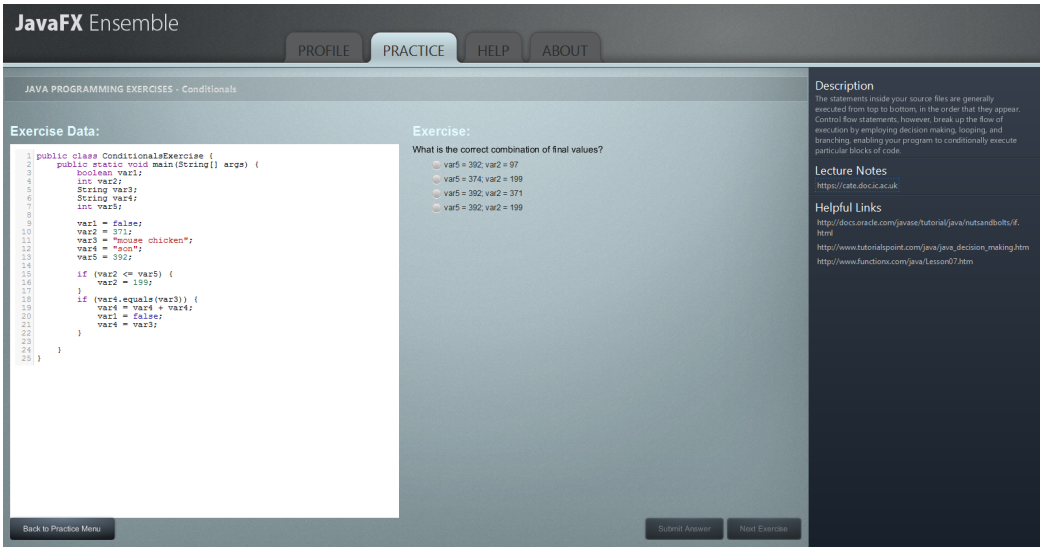


Figure 4.11: The Practice tab view showing an exercise on conditionals.

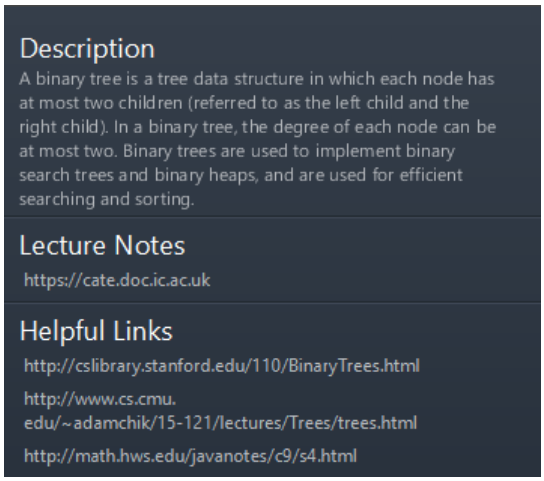


Figure 4.12: An example sidebar in the Practice tab.

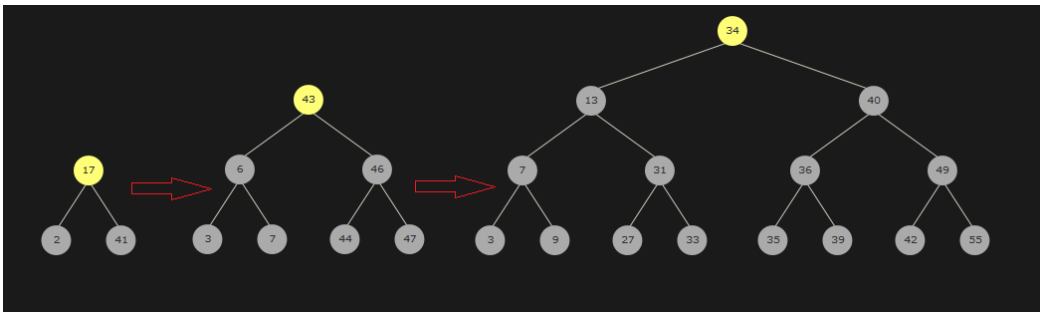


Figure 4.13: Progression of binary tree exercises.

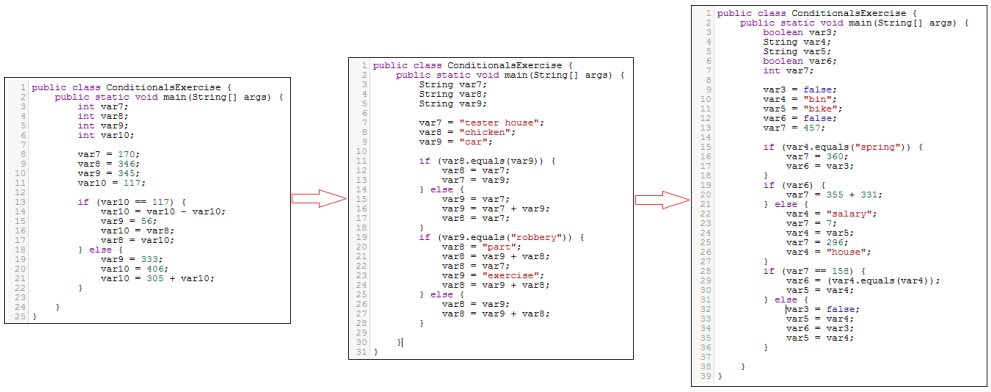


Figure 4.14: Progression of conditionals exercises.

4.2 Teacher view

The teacher's main access to the system is through the jSCAPE admin tool. Essentially it provides an interface to the database where all information about students, performance, exercise categories and exercises is stored. This tool was developed to allow teachers, not familiar with SQL, to still be able to retrieve useful data about students in a presentable way and to facilitate management of the exercise bank.

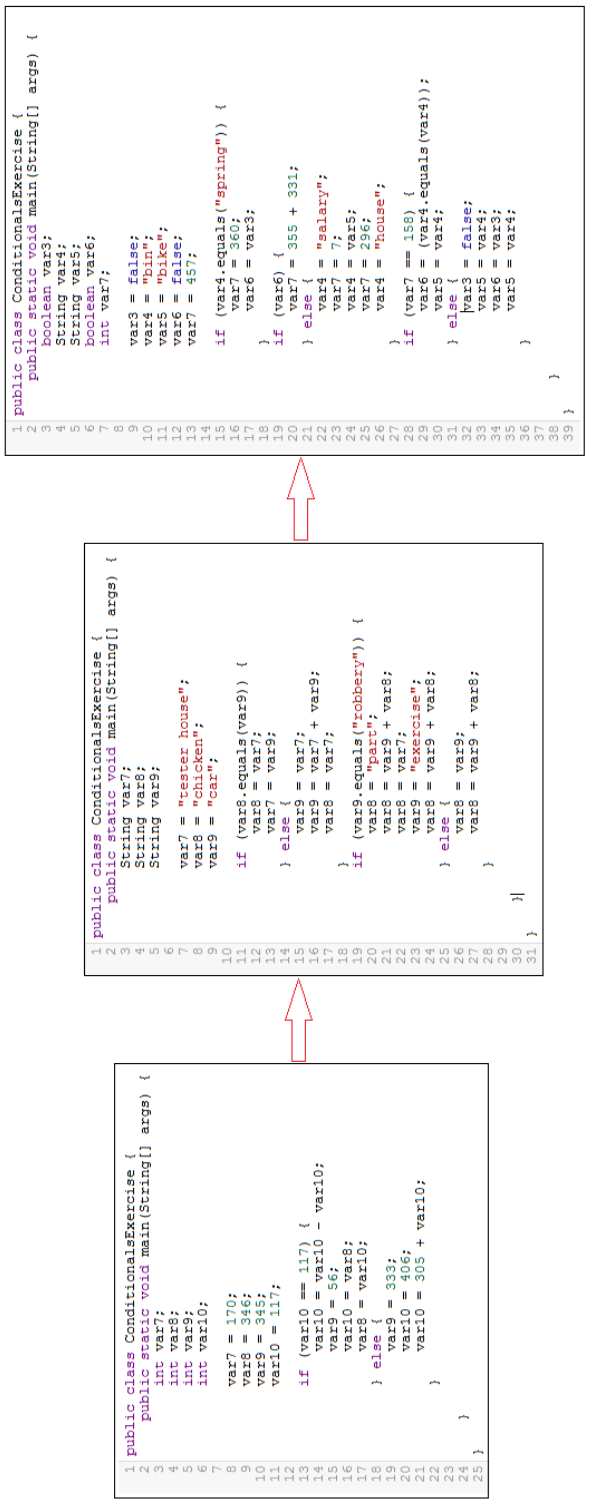


Figure 4.15: Progression of conditionals exercises.

4.2.1 Tracking student progress

Figure 4.16: An overview of the Analyze tab in the jSCAPE admin tool.

The jSCAPE admin tool provides teachers with the ability to track student progress and performance over time. Figure 4.16 gives an overview of the Analyze tab, where statistical data about selected students can be displayed. The information displayed is identical to that displayed in the jSCAPE Profile tab (section 4.1.2). On the left hand side of the window, the light blue box contains options to filter which data is displayed in the main window. A close up of the filter options is shown in figure 4.17.

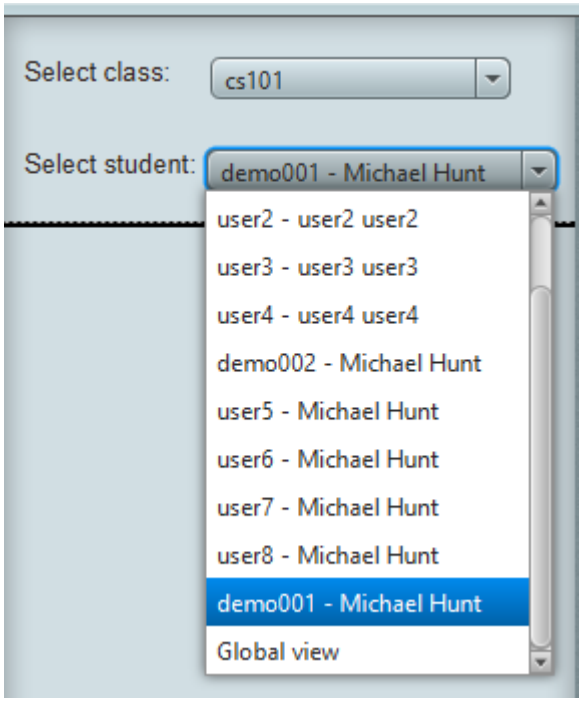
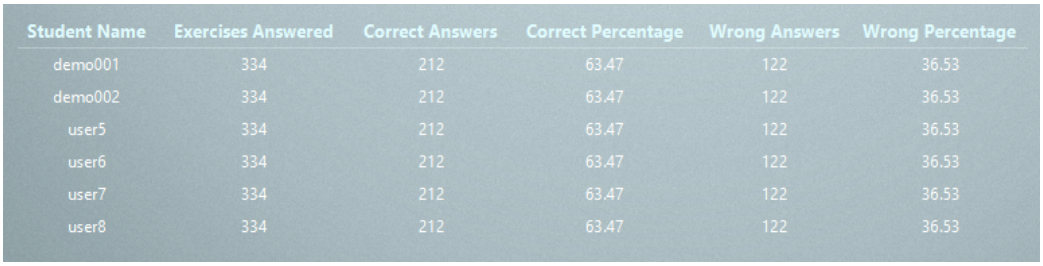


Figure 4.17: Selection possibilities in the Analyze tab.

The jSCAPE system includes support for multiple classes to allow for both the separation of students and the separation of exercises available to a class. A teacher can select a class to view statistics about those students taking the class. This will update the list of students in the combo box, allowing the teacher to focus his attention on the performance of one particular student.

Selecting a student will show their profile information in the light blue window, along with the date of their last login, and the date of their last exercise answered. In addition, the pie charts, performance table and progress graphs will be updated to reflect the performance of the selected student. Finally, there is an option to obtain a global view of the class' performance by selecting the "Global view" option.



Student Name	Exercises Answered	Correct Answers	Correct Percentage	Wrong Answers	Wrong Percentage
demo001	334	212	63.47	122	36.53
demo002	334	212	63.47	122	36.53
user5	334	212	63.47	122	36.53
user6	334	212	63.47	122	36.53
user7	334	212	63.47	122	36.53
user8	334	212	63.47	122	36.53

Figure 4.18: Global statistics view of a class.

Figure 4.18 shows the table that is displayed after selecting the global view option. This table shows all the students who have answered exercises in a particular exercise category. In the example above, the data shown is for the "Syntax" exercise category. The student user names are listed along with the number of exercises they have answered, and a detailed breakdown of the number of correct and wrong answers in terms of raw values and percentages.

There is a combo box to select which exercise category to display, but this isn't shown in the picture to minimize the size of it. The global view feature is useful for teachers to identify which students may be facing difficulties. They can then select the student in the Analyze tab to get more detailed statistics and information about the student's progress.

4.2.2 Managing the exercise bank

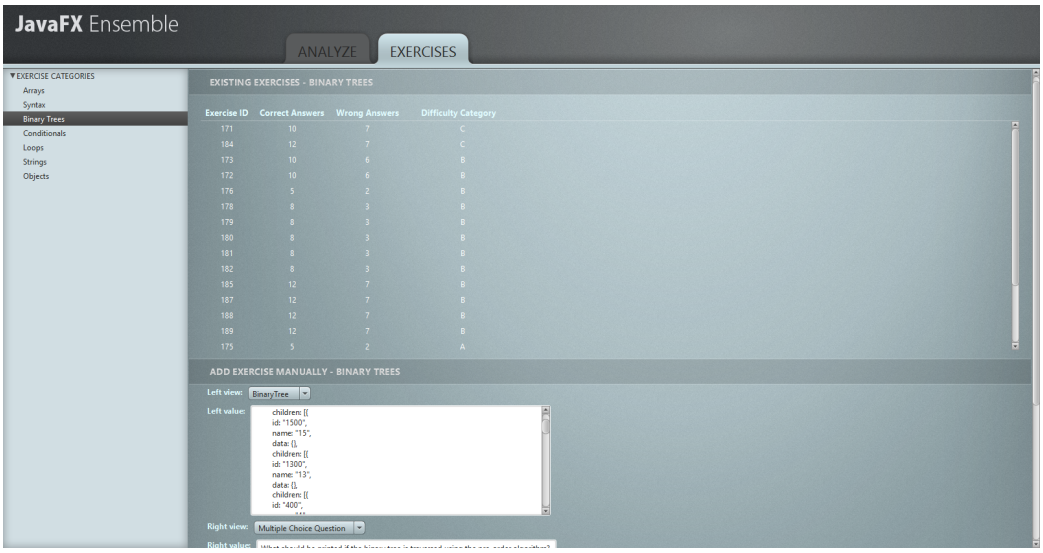


Figure 4.19: An overview of the exercise bank management tab.

Managing exercise categories

add description, lecture links and helpful links
set exercise category to not visible, useful feature if the class isn't ready to answer exercises of this category because the relevant material hasn't been taught yet, but the teacher still wants to prepare the exercises in advance to save time.

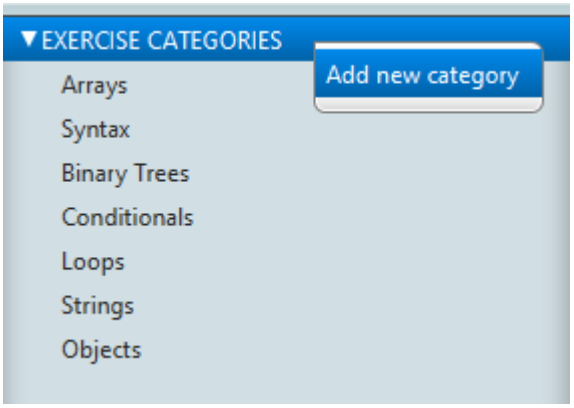


Figure 4.20: Adding a new exercise category.

Viewing existing exercises

EXISTING EXERCISES - BINARY TREES

Exercise ID	Correct Answers	Wrong Answers	Difficulty Category
171	10	7	C
173	10	6	B
175	5	2	A
172	10	6	B
174	5	2	A
176	5	2	B
177	5	2	A
178	8	3	B
179	8	3	B
180	8	3	B
181	8	3	B
182	8	3	B
183	8	3	A
184	12	7	C
185	12	7	B

Figure 4.21: Viewing information about existing exercises.

Adding an exercise manually

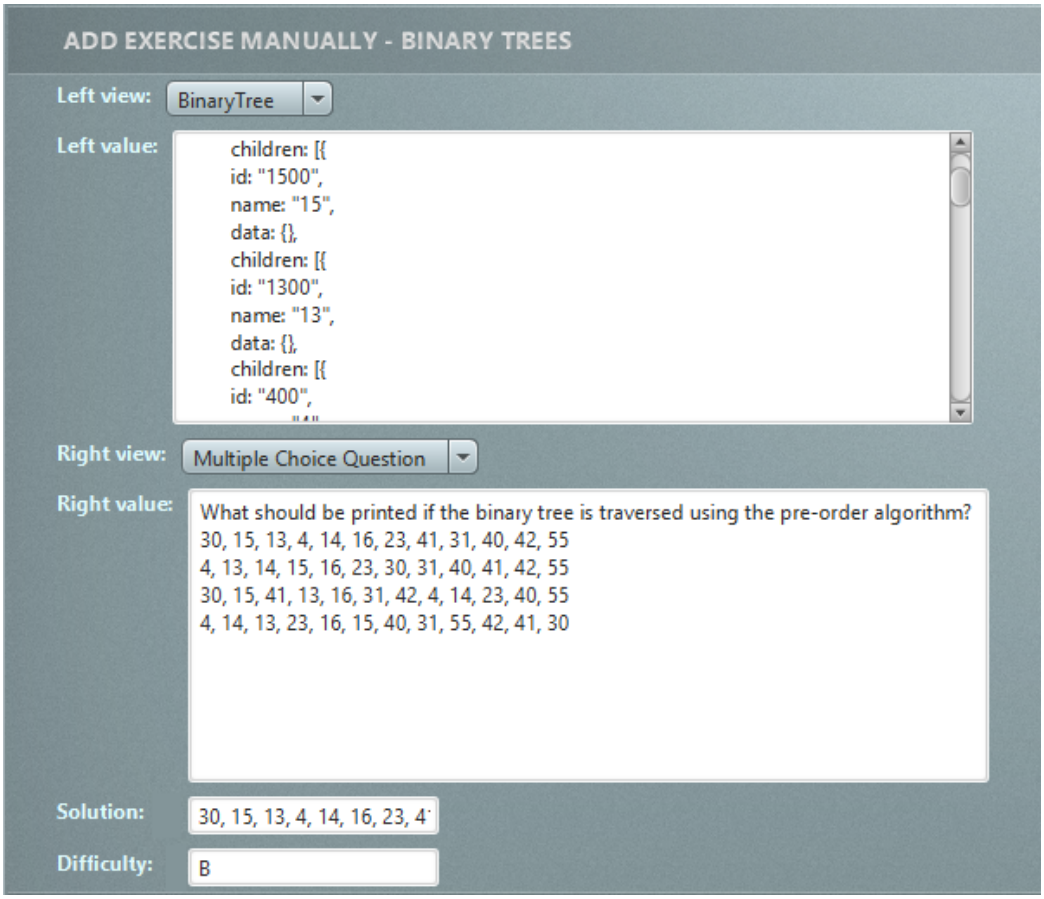


Figure 4.22: Adding an exercise on binary trees manually.

Automatically generating exercises

runs the appropriate exercise generator and stores the exercises in the database.

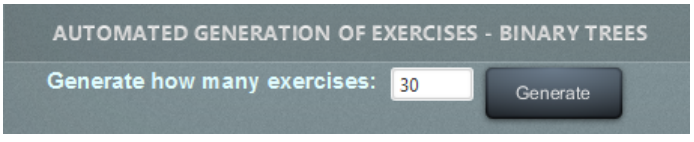


Figure 4.23: Automatically generating a number of new exercises for a specific exercise category.

4.3 Summary

In this section we gave an overview of the components present in the jSCAPE system.
We showed that jSCAPE performed a lot of tracking of student's performances by storing useful statistics concerning their progress. In addition

In the following chapter, we talk more about the design of the system and various interesting implementation details and difficulties faced during the development of this project.

Chapter 5

Design and Implementation

Talk about design choices such as only multiple choices, no exercises asking to write code, writing custom server, etc...
Mention three tier architecture
implemented as a JavaFx applet
list tools+technology and evaluate advantages/disadvantages
java programming exercises
server implementation, message codes, objectin/out streams, serverthread
CAT development, we refer back to the five components of a CAT...what item selection algorithm we use, what scoring procedure, no termination criterion, entry point is average knowledge distribution initially and attempts at a calibrated item pool, currently with teacher providing the parameters since obtaining a high quality calibrated item pool isn't something I can do.

```
1  /**
2   * Calculates the item information for the provisional proficiency level
3   * estimated until that moment, i.e. thetaEstimate.
4   */
5  private double itemInformation(int thetaEstimate, Item item) {
6      double information;
7
8      double a = item.getA();
9      double c = item.getC();
10
11     double aSquared = Math.pow(a, 2);
12     double p = probabilityCorrectAnswer(thetaEstimate, item);
13     double q = 1 - p;
14
15     information = aSquared * (q / p);
16
17     double numerator = p - c;
18     double denominator = 1 - c;
19
20     information = information * Math.pow(numerator / denominator, 2);
21
22     return information;
23 }
```

Listing 5.1: Item information algorithm.

```
1  /**
2   * Calculates P(correct|theta), i.e. the probability of a correct answer
3   * given an ability level of theta.
4   */
5  private double probabilityCorrectAnswer(int theta, Item item) {
6      double a = item.getA();
7      double b = item.getB();
8      double c = item.getC();
9
10     double probability = (1 - c) / (1 + Math.exp(-1.7 * a * (theta - b)));
11     probability = c + probability;
12
13     return probability;
14 }
```

Listing 5.2: Item response function algorithm.

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Listing 5.3: Example exercise illustrating the XML format.

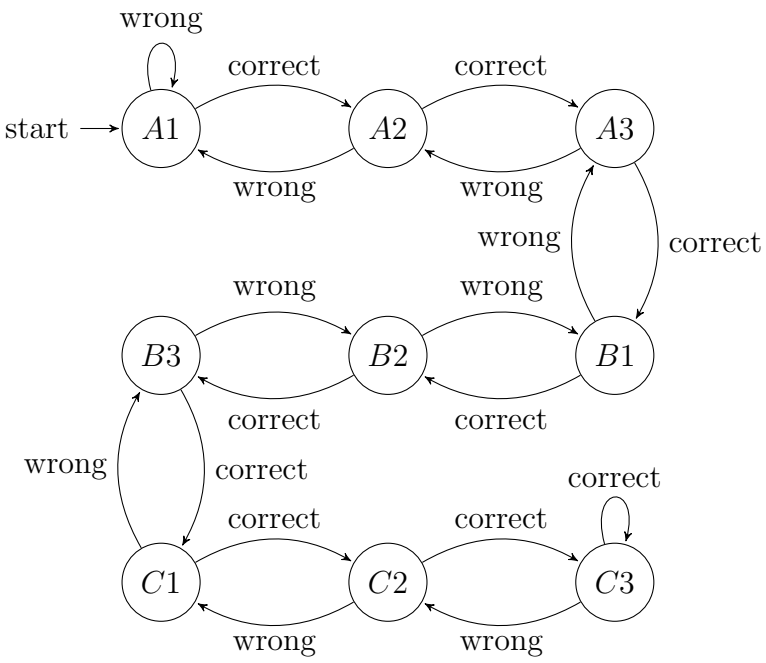


Figure 5.1: State machine of adaptive difficulty categories.

Chapter 6

Evaluation

Mention how our developed system performs against the advantages and disadvantages of CAT.
The evaluation stage will address mostly these two aspects:

- The system has correctly modelled the ability of students
- The system is useful in helping students to learn programming and helping lecturers with getting feedback on their teaching, in the form of statistics.

6.1 Qualitative

- Surveys to get feedback from students on interface, usability, etc...
-

6.2 Quantitative

- Statistical analysis to evaluate item calibration and modelling of student's abilities
-

evaluate the fact that teachers have to enter the parameters of items.

Chapter 7

Conclusion

7.1 Future Work

We have a few ideas of where to orient the project next...

- Very flexible system so other programming languages could be offered, i.e. cSCAPE, for C and hSCAPE for Haskell.
- Working more extensively on the adaptive component of the system, i.e. improving the algorithm which selects questions for students based on their estimated ability.
- Extend system to allow admins to provide their own question templates, maybe come up with a template grammar which then allows questions to be automatically generated. Or at least allow pluggable function references which will be called to generate the exercise component.
- Add support for more question types. JavaFX is very good in that sense since it can play audio clips, video clips, show animations, the webview component has endless possibilities thanks to the inclusion of Javascript.
- Future Work as a research project vs future work as a commercial product
- JavaFX applications are based on the model-view-controller pattern, so a nice split can be done in the code, however I only learnt about this two weeks into the project, so all the of the GUI components are created in the code as opposed to in the FXML file.

- add more robust security: hashing for login/password and ssl connections between the server and client, and server database, because this communication could reveal solutions to the exercises..

Bibliography

[1] Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-de-la-Cruz, J. L., & Rios, A. (2004). SIETTE: A Web-Based Tool for Adaptive Testing. *International Journal of Artificial Intelligence in Education, 14*, 29-61.

[2] Computerized adaptive testing. `http://en.wikipedia.org/wiki/Computerized_adaptive_testing`. Accessed: June 12, 2014.

[3] Thompson, Nathan A., & Weiss, David A. (2011). A Framework for the Development of Computerized Adaptive Tests. *Practical Assessment, Research & Evaluation*, 16(1).

[4] IRT-Based CAT. `http://www.iacat.org/content/irt-based-cat`. Accessed: June 12, 2014.

[5] Weiss, D. J., & Kingsbury, G. G. (1984). Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, 21, 361-375

[6] Weiss, D.J. (1985). Adaptive Testing by Computer, *Journal of Consulting and Clinical Psychology*. 1985, 53, 6, pp. 774-789.

[7] Wainer, H., & Mislevy, R.J. (2000). Item response theory, calibration, and estimation. In Wainer, H. (Ed.) Computerized Adaptive Testing: A Primer. Mahwah, NJ: Lawrence Erlbaum Associates.

[8] Item Response Theory. `http://en.wikipedia.org/wiki/Item_response_theory`. Accessed: June 12, 2014.

[9] Baker, Frank (2001). The Basics of Item Response Theory. (2nd Edition). Available at `http://info.worldbank.org/etools/docs/library/117765/Item%20Response%20Theory%20-%20F%20Baker.pdf`. Accessed: June 12, 2014.

[10] Partchev, Ivailo (2004). A visual guide to item response theory. Available at `www.metheval.uni-jena.de/irt/VisualIRT.pdf`. Accessed: June 12, 2014.

[11] Chatzopoulou, D. I. & Economides, A. A. (2010). Adaptive assessment of student's knowledge in programming courses. *Journal of Computer Assisted Learning*, Vol. 26, No. 4.

Appendix A

User Manual