



londonknowledgelab

exploring the future of learning with digital technologies



Birkbeck
UNIVERSITY OF LONDON

Towards a Formalization of the Automatic Generation of Exercises

Sergio Gutiérrez, Francisco J. Losa and Carlos Delgado Kloos
sergut@lkl.ac.uk, {fjlosa,cdk}@it.uc3m.es

Table of Contents

- Introduction: Automatic Exercise Generation
- Different approaches
- The need of parameters
 - Four different types of parameters
 - Examples
- Prototype
- Conclusions

Introduction

- Almost any EAH system needs exercises
 - Supporting the understanding of difficult concepts
 - Specific pedagogical approaches: learning-by-doing, problem-based learning, etc
 - Testing the knowledge of the learner: self-testing or assessment
 - Providing information to related processes: user modelling, sequencing, etc
- Questions and exercises are part of almost any learning process

Introduction: the more, the merrier

- When it comes to exercises, the higher the number, the better
 - A small number of different exercises might be succeeded by the learners without having really mastered the concepts behind (gaming the system?)
 - A higher number of exercises allows for a higher level of adaptation (e.g. selecting a level of difficulty, adapting the sequence, etc)
- In an ideal world, we would have an infinite number of exercises

Introduction

- However, creating exercises takes time
- Correcting exercises takes even more time
- In an ideal world:
 - Exercises would be created automatically (different exercises)
 - Exercises would be marked automatically

The problem

- Is it possible to create a framework for the automatic generation of (EAH-) exercises that:
 - is domain-independent?
 - does not require a high technological background?

Some approaches

- Ad-hoc solutions
 - Creation of exercises based on some specific rules of the domain
 - Creation of exercises according to some heuristics of the model
 - Common approaches, but difficult to generalise

Some approaches

- Formula-based exercise generation
 - The answer is found by solving an equation
 - The system selects most parameters and asks for the missing one
 - Marking by value substitution in the equation
 - Only adequate for domains like maths or physics

Some approaches

- Programming exercises
 - Use of a real compiler for detecting errors
 - The use of a general-purpose tool for the correction of the learners' solutions lowers the cost of creating multiple exercises
 - Difficult to generalise this approach beyond programming

Some approaches

- IMS Question and Test Interoperability
 - The IMS specification describes several types of exercises
 - The exercises do not vary, they are always the same
 - Big success in systems interoperability, but no adaptation/modification of exercises
 - Is it possible to have an adaptive QTI?

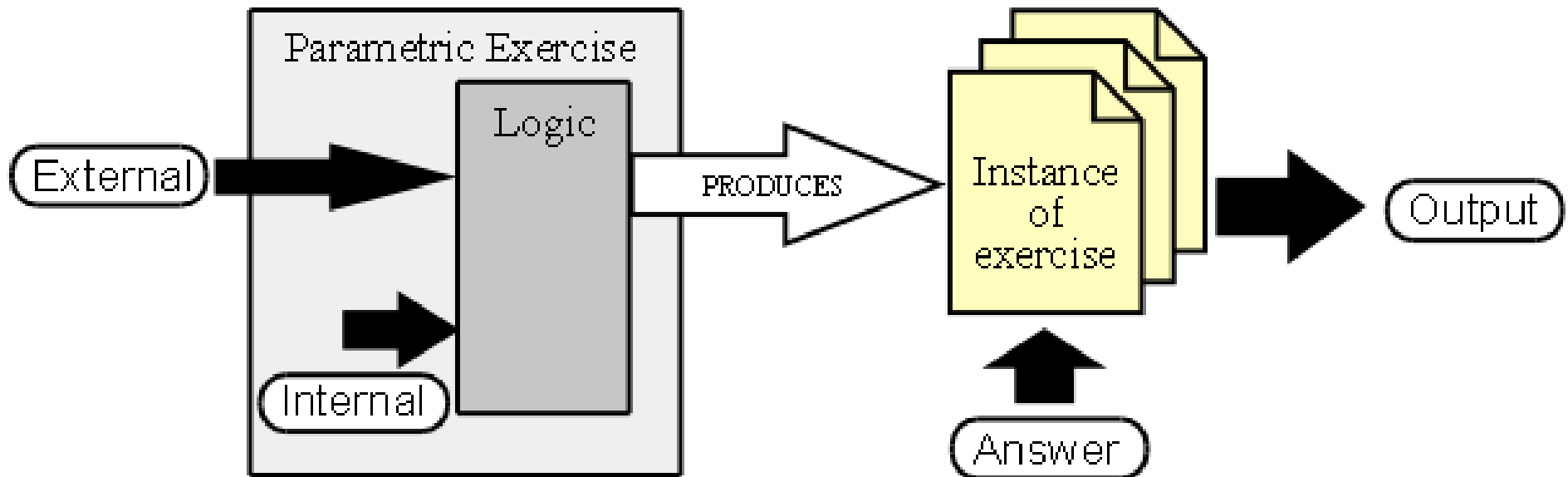
Parametric Exercises

- Many approaches to automatic generation of exercises rely in some form of parameter
- Parameters modify the creation of the exercise: every instance is a new exercise
- Parameters are used for marking (if any)
- Is it possible to formalize the idea of parametric exercise (parametric QTI?)?
- First step: what kind of parameters are there?

Parameters

- We have identified four types of parameters
 - External (input)
 - Internal (input)
 - Answer
 - Output

Parameters



Example 1

- Computer Architecture: Hard-disk read request (scheduling)
- Answers are text-based
- Parameters:
 - External: read requests, disk size, scheduling algorithm, etc
 - Internal: seed for randomizing unspecified external parameters
 - Answer: total distance travelled by disk head
 - Output: time needed, correct answer

Example 2

- Geography: position of countries and cities
- Answers are graphic-based (click or drag)
- Parameters:
 - External: continent, number of countries
 - Internal: none
 - Answer: positions
 - Output: total time, answers that needed more time, answer that were answered quickly, ratio of correct answers

Our implementation

- We have implemented a web-based parametric exercise generator
- It allows the designer to specify the parameters, the layout of the exercise and how to mark it
- Some feedback can be given depending on the answers

Our implementation

The screenshot shows a web browser window titled "Parametric Exercises Editor - Mozilla Firefox". The address bar shows the URL "http://localhost:8180/pared/ParEdClient.html". The browser's menu bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The toolbar shows various navigation icons and several open tabs, including "tomcat 5.5 u...", "Formatter (Ja...", "VerticalPanel", "Java 2 Platfor...", "Parametr...", "ParEx - Para...", "CSS Text Pro...", and "Gmail - Poste...".

The main content area is titled "Parametric Exercises Editor" and features a logo on the left and a printer icon on the right. Below the title is a navigation bar with "Edit" and "Validate/Preview" options. The "Edit" option is selected, and the page is divided into three sections:

- Parameter Input Section:** A table with columns for "Parameter Name", "Data Type", "Parameter Type", and "Default Value". Each row includes a "remove" link.
- Exercise Description Section:** A text area containing HTML code for an exercise description.
- Feedback Section:** A section for user feedback.

The status bar at the bottom shows "Done" and a green checkmark icon.

Parameter Name	Data Type	Parameter Type	Default Value	
seed	Float	External	\$random	remove
numBytes	Integer	External	2	remove
sourceCodif	Integer	Internal		remove
destCodif	Integer	Internal		remove
mode	String	External		remove
ans	String	Answer		remove
solving_time	String	Output	\$time	remove
qualification	String	Output		remove

[add parameter](#)

Exercise Description Section

```
<p> Now, in order to increase a little bit the exercise difficulty, we will create a multi-format number converter. </p>

<p> For example, let's try to convert the following expression from $sourceCodif to $destCodif: </p>

[CONT class=NumberFormatConverterContainer]

<p> Now, please type here your answer: [TEXT var=ans] </p>

<p>Good Luck!</p>
```

Feedback Section

Our implementation

Parametric Exercises Editor - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

http://localhost:8180/pared/ParEdClient.html

ParEd: Web-based Parametric Exercise Editor

File Edit Validate/Preview

Parameter Input Section

Parameter Name	Data Type	Parameter Type	Default Value
	String	External	<input type="text"/> remove

[add parameter](#)

Exercise Description Section

This variable does not exist: \$var1

Feedback Section

This one neither: \$var2

▼ [Define exercise name and add attachments \(click to hide/expand\)](#)

Exercise name (only alphanumeric characters):

Container widget (.java file): Examinar...

Exercise Logic (.java file): Examinar...

Terminado

The following errors were found. Please correct them before proceeding:

Parameter Input section errors - parameter definitions:

- Parameter number 1 cannot be left blank. Please introduce a parameter name.
- There is no Output type parameter. There must be at least one in order to send it at the end of the exercise

'Exercise Description' section errors:

Internal references error

- Parameter '\$var1' is undefined in the "Input Parameters" section. Please define it first before referencing to it.

'Feedback' section errors:

Internal references error

- Parameter '\$var2' is undefined in the "Input Parameters" section. Please define it first before referencing to it.

Our implementation

ParEd: Web-based Parametric Exercise Editor - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

http://localhost:8180/pared/ParEdClient.html

ParEd: Web-based Parametric Exercise Editor

File Edit Validate/Preview

seed: random
min_n_cylinders: 150
max_n_cylinders: 200
n_cylinders:
min_n_requests: 4
max_n_requests: 12
n_requests:
algorithm:
initial_position:

Test values

Terminado

Imagine that we have a hard disk unit with $n=197$ cylinders, numbered from 0 to $n-1$, and a set of access requests is received as displayed in List 1.

Knowing that the disk access scheduling algorithm in use is *SSTF (Shortest Seek Time First - on equal distance suppose that the head moves in ascendent direction)*, and that the reading head is initially positioned in cylinder number 124, please indicate in List 2 the order in which the requests would be processed.

List 1: 57 149 158 56
List 2:

Incorrect. The correct sequence is: 149, 158, 57, 56

Moreover, which would be the total distance (measured in number of cylinders) that the reading head moves to process all requests?

Incorrect. The correct answer is: 136

Exercise Feedback

Here goes some feedback about disk access scheduling algorithms

Our implementation

- We have implemented a web-based parametric exercise generator
- It allows the designer to specify the parameters, the layout of the exercise and how to mark it
- Some feedback can be given depending on the answers
- Main limitation: complex exercises need to be programmed

Conclusions

- EAH systems would benefit from a standard way for generating/marking different exercises
- Most approaches to automatic exercise generation are difficult to generalise
- The four basic types of parameters have been identified
- Next step: overcome the 'programming' restriction of our prototype

Thank you very much!
Questions?