Design and Implementation of OpenDSA Interoperable Infrastructure

Hossameldin Shahin, Master Defense
Computer Science, Virginia Tech
June 30, 2017
Blacksburg, Virginia
Outline

• Background
  ○ The First Generation OpenDSA Infrastructure
• Original Infrastructure: Challenges, Limitations, and Solutions
• OpenDSA-LTI
  ○ From a Monolithic System to an Interoperable Infrastructure
• Contributions
• Future Work
Background: OpenDSA Infrastructure

1. OpenDSA content
   ○ The prose → ReST files
   ○ Visualizations and Exercises → HTML5, CSS, and JavaScript

2. Compilation process
   ○ Generates complete books from content types

3. Data Collection Server (DCS)
   ○ Tracks student registrations, interaction data, and scores

4. Client-side framework
   ○ Managing the communication between the compiled HTML pages and the DCS
First Generation OpenDSA infrastructure
1- Original Infrastructure: Challenges, Limitations, and Solutions
Original Infrastructure Challenges

1. OpenDSA content
   - Outdated Khan Academy exercises framework
   - The “gaming” problem

2. Client-side framework
   - Buffering mechanism → Missing interaction data
   - Caching mechanism → Out-of-sync and gradebook problems

3. Compilation process
   - Manually prepared configuration files.

4. Data Collection Server (DCS)
   - Security concerns
   - Unstable automated assessment engine for programming exercises
   - No support for additional exercise types (e.g. CodeWorkout)
Original Infrastructure Challenges

1. OpenDSA content
   ○ Outdated Khan Academy exercises framework
   ○ The “gaming” problem

2. Client-side framework
   ○ Buffering mechanism → Missing interaction data
   ○ Caching mechanism → Out-of-sync and gradebook problems

3. Compilation process
   ○ Manually prepared configuration files.

4. Data Collection Server (DCS)
   ○ Security concerns
   ○ Unstable automated assessment engine for programming exercises
   ○ No support for additional exercise types (e.g. CodeWorkout)
1. Mitigating Khan Academy Problems

KA summary exercise

A range query is:

- A way of efficiently locating the records in a database
- Where a record is returned if its relevant key value falls between a pair of values
- Where a record is returned if its unique identifier matches the search value
- Another name for any database query for a key value
1. Mitigating Khan Academy Problems (Cont.)

**Technical problem:** Outdated framework version

1. **Update or replace KA framework?**
   - Update the framework
   - Implement OpenDSA requirements as a layer that overrides some of the KA framework functionalities

2. **Latest KA uses RequireJS library**
   - Enhance user experience by reducing loading time
   - Help developers by separating JS code from HTML files

3. **Updated framework introduces new limitation!**
   - No native method to recursively load Summary exercises.
1. Mitigating Khan Academy Problems (Cont.)

**Pedagogical problem:** Students found ways to fool the framework

1. **How students game the system?**
   - Continuous refreshes to get an easy question
   - Misuse the hint system

2. **Preventing the gaming behavior**
   - Remember the “current” question
   - Remember when hint system is used
   - Remove correctly answered questions

3. **Proper handling for some use cases**

4. **Preventing gaming has direct impact on OpenDSA research**
Original Infrastructure Challenges

1. OpenDSA content
   ○ Outdated Khan Academy exercises framework
   ○ The “gaming” problem

2. Client-side framework
   ○ Buffering mechanism → Missing interaction data
   ○ Caching mechanism → Out-of-sync and gradebook problems

3. Compilation process
   ○ Manually prepared configuration files.

4. Data Collection Server (DCS)
   ○ Security concerns
   ○ Unstable automated assessment engine for programming exercises
   ○ No support for additional exercise types (e.g. CodeWorkout)
2. Mitigating Client-Side Framework Problems

1. Missing interaction data
   ○ Two students use the same browser
   ○ Students close their browser abruptly
   ○ Wrong mechanism for communicating interaction data

2. An out-of-synch local proficiency cache
   ○ Define the DCS as the ground truth for student grades
   ○ Side effect, Internet connection is required all the time
Original Infrastructure Challenges

1. OpenDSA content
   ○ Outdated Khan Academy exercises framework
   ○ The “gaming” problem
2. Client-side framework
   ○ Buffering mechanism → Missing interaction data
   ○ Caching mechanism → Out-of-sync and gradebook problems
3. Compilation process
   ○ Manually prepared configuration files.
4. Data Collection Server (DCS)
   ○ Security concerns
   ○ Unstable automated assessment engine for programming exercises
   ○ No support for additional exercise types (e.g. CodeWorkout)
3. Mitigating Compilation Process Problem

- Config files has to be aligned with the ReST files
  - Manual, error prone process
  - The content keeps changing
  - Multiple config files need to be changed.

Solution: Automatically extract all the config files from the ReST files.
Original Infrastructure Challenges

1. **OpenDSA content**
   - Outdated Khan Academy exercises framework
   - The “gaming” problem

2. **Client-side framework**
   - Buffering mechanism → Missing interaction data
   - Caching mechanism → Out-of-sync and gradebook problems

3. **Compilation process**
   - Manually prepared configuration files.

4. **Data Collection Server (DCS)**
   - Security concerns
   - Unstable automated assessment engine for programming exercises
   - No support for additional exercise types (e.g. CodeWorkout)
4. Data Collection Server

1. Proprietary mechanism for handling session keys
   - Modern Web framework provides full support for session handling
   - The secure approach is to use cookies to handle session data
   - Unfortunately, DCS didn’t use the standard, secure approach
   - Using proprietary mechanism:
     i. Insecure communication with client-side framework
     ii. Allowed only one session key per user
     iii. Complicated client-side framework code

2. Automated assessment engine for programming exercises
   - DCS contains embedded engine to store and evaluate students’ code
   - Students complained about slowness and instability

3. No support for additional exercise types (e.g. CodeWorkout)
Monolithic System vs. Learning Tool

1. Original infrastructure
   ○ DCS needs to be replaced

2. OpenDSA is a monolithic system
   ○ Students need to remember multiple usernames/passwords
   ○ Track their progress in multiple systems
   ○ Proprietary instructor tools (the gradebook)
   ○ Instructors have to create and maintain the same course across several systems.

3. OpenDSA as a learning tool
   ○ Keep instructors and students in one learning environment
   ○ Deliver OpenDSA through a trusted channel
   ○ Take advantage of the LMS robust instructor’s tools
OpenDSA Interoperability

Standard integration protocol

Ruby, Python
PHP, Java
2- OpenDSA-LTI for Interoperability
Interoperability Alternatives

1. OpenDSA requirements and constraints
   ○ Modifications required to OpenDSA content code base
   ○ The ability to integrate OpenDSA with a range of LMSs
   ○ Ability to augment OpenDSA with other learning tools (such as new exercise types) within an LMS
   ○ Community support and evidence for continuity

2. Integration approaches
   ○ A Component-Oriented Approach
   ○ Plug-In Architecture
   ○ Widgets
   ○ SCORM
   ○ IMS Learning Tools Interoperability (LTI)
How LTI Protocol Works

1. Terminology
   ○ Tool consumer/provider

2. IMS LTI v1.1 allows:
   ○ The outcomes services
   ○ IMS Learning Information Services (LIS)

3. Before the launch process
   ○ The Admin has to define the tool provider in the LMS
   ○ The instructor creates an activity that refers to the learning tool
OpenDSA-LTI launch function sequence diagram

1. Student requests a page
2. Response with external tool link (hidden form + Javascript)
3. LTI Launch message (auto submitted form)
4. iframed resource loaded
• **Devise**: Authentication
• **CanCanCan**: Authorization
• **Activeadmin**: Admin user interface
Identity Management

1. Authentication
   ○ Student’s automatic registration process
   ○ Instructor's manual registration, permission request, and access token update
   ○ Third-party authentication mechanisms

2. Authorization
   ○ Two roles permission model (global and course roles)
   ○ Instructor's privileges
   ○ OpenDSA-LTI Admins
   ○ Students are not aware of OpenDSA-LTI
User Interfaces

1. Instructor Interfaces
   ○ OpenDSA-LTI interface to configure and compile books within the LMS

2. Student Interfaces
   ○ The LMS interface

3. Admin Interfaces
   ○ OpenDSA-LTI provides comprehensive admin interfaces

Bootstrap library is used for a responsive and fluid layout. It scales to different screen sizes and resolution.
New Course Offering

All fields are required.

Organization: Virginia Tech

Course: CS314: Data Structures & Algorithms

Term: Spring 2017

Label: TR 10:00am

Write a label for each course section.

Book Instance: CS3 Data Structures & Algorithms

Canvas Instance: https://canvas.instructure.com

Canvas access token: 7-rVi4J6xw/6MKpUy9B8Ep5xhlIlYuVlts8oP13sEz2mF9P8EdxvO7Qqk15CZ

Put your Canvas access token for the selected Canvas instance. If you have not already created a Canvas access token, then follow the instructions here to make one.

Canvas course Id: 113954

Create a new course at the selected canvas instance and copy the course id here (e.g. Course id of https://canvas.instructure.com/courses/1076903 is 1076903).

Submit
Instructors Interfaces: Book configuration 2/3
1. Individual visualizations instead of a complete book.
2. Canvas implemented an extension to LTI v. 1.1 for resource selection.
3. The extension became part of v. 2.0.
4. Instructors have to manually define an LTI app in the course to be able to use this feature.
5. This feature still under active development.
Students Interface: Main navigation page 1/2
Example 1.1.2

A company is developing a database system containing information about cities and towns in the United States. There are many thousands of cities and towns, and the database program should allow users to find information about a particular place by name (another example of an exact-match query). Users should also be able to find all places that match a particular value or range of values for attributes such as location or population size. This is known as a **range query**.

A reasonable database system must answer queries quickly enough to satisfy the patience of a typical user. For an exact-match query, a few seconds is satisfactory. If the database is meant to support range queries that can return many cities that match the query specification, the user might tolerate the entire operation to take longer, perhaps on the order of a minute. To meet this requirement, it will be necessary to support operations that process range queries efficiently by processing all cities in the range as a batch, rather than as a series of operations on individual cities.

The hash table suggested in the previous example is inappropriate for implementing our city database, because it cannot perform efficient range queries. The B*-tree supports large databases, insertion and deletion of data records, and range queries. However, a simple **linear index** would be more appropriate if the database is created once, and then never changed, such as an atlas distributed on a CD or accessed from a website.

1.1.1.4. Introduction Summary Questions

Which of these is NOT a definition for efficiency in a computer program?

- It solves the problem within the required resource constraints
- It requires fewer resources than known alternatives
- It runs in linear time

**Answer**: It requires fewer resources than known alternatives

**Need help?**

Follow a link
Recent Errors

<table>
<thead>
<tr>
<th>CLASS NAME</th>
<th>MESSAGE</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoMethodError</td>
<td>undefined method # for null:NilClass</td>
<td></td>
</tr>
</tbody>
</table>

Current Term

<table>
<thead>
<tr>
<th>NAME</th>
<th>STARTS ON</th>
<th>ENDS ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2017</td>
<td>January 10, 2017</td>
<td>May 15, 2017</td>
</tr>
</tbody>
</table>

Spring 2017 Course Offerings

<table>
<thead>
<tr>
<th>COURSE</th>
<th>OFFERING</th>
<th>INSTRUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp 232: Data Structures and Problem Solving</td>
<td>TR 2:00</td>
<td>Taylor Rudahl</td>
</tr>
<tr>
<td>SDAP13: SDAP13</td>
<td>TR 2:00</td>
<td>Taylor Rudahl</td>
</tr>
<tr>
<td>CS3114: Data Structures &amp; Algorithms</td>
<td>TR 10:00</td>
<td>Hosseinedin Shahin</td>
</tr>
</tbody>
</table>

Powered by Active Admin 1.0.0.pre1
Security

1. Students personal and class performance information are considered very sensitive (FERPA).
2. Security solutions such as SSL and firewalls are insufficient in protecting Web applications.
3. Top 10 threats, Open Web Application Security Project (OWASP)

- Injection
- Broken Authentication & Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Cross-Site Request Forgery (CSRF)
- Insecure Cryptographic Storage
- Failure to Restrict URL Access
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards
Contributions

1. Make OpenDSA more robust system
2. Integrate OpenDSA with the widest range of LMSs.
3. The OpenDSA-LTI makes it easier for instructors and students to use OpenDSA contents.
4. Transform OpenDSA from a monolithic system to a learning tool.
5. The OpenDSA-LTI infrastructure:
   ○ Hosts **36 active courses** offered by 25 different universities in 6 countries.
   ○ Used by **41 instructors** and **2,722 registered students.**
Future Directions

- Learning analytics is important to improve student performance and predict future outcomes
- Instructors can design all their activities to happen inside the LMS
- However, activities would be scattered across a range of tools
- Scattered learning tools means incompatible interaction capture mechanisms and data formats. (OpenDSA-LTI is an example)
- Caliper is a standard that enables interaction data with learning tools and LMSs to be collected, stored, and transported
- Make OpenDSA and encouraging other teams in the field to make more Caliper-compliant tools
Publications

Acknowledgements

- Dr. Clifford A. Shaffer
  - Advisor and committee chair
- Dr. Edward A. Fox and Dr. Dennis Kafura
  - Committee members
- OpenDSA and CodeWorkout developers
- NSF
  - Grant DUE-1432008
- My wife Doaa, and my kids Zeyad and Somaia
Thank you!
Questions?