A Fully-Autonomous Centralized ERP Application for Teaching Higher-Education Courses

(CIS601 - Graduate Seminar 12/04/17)

Daniel Izadnegahdar CLEVELAND STATE UNIVERSITY

Dr.Sunnie Chung CLEVELAND STATE UNIVERSITY

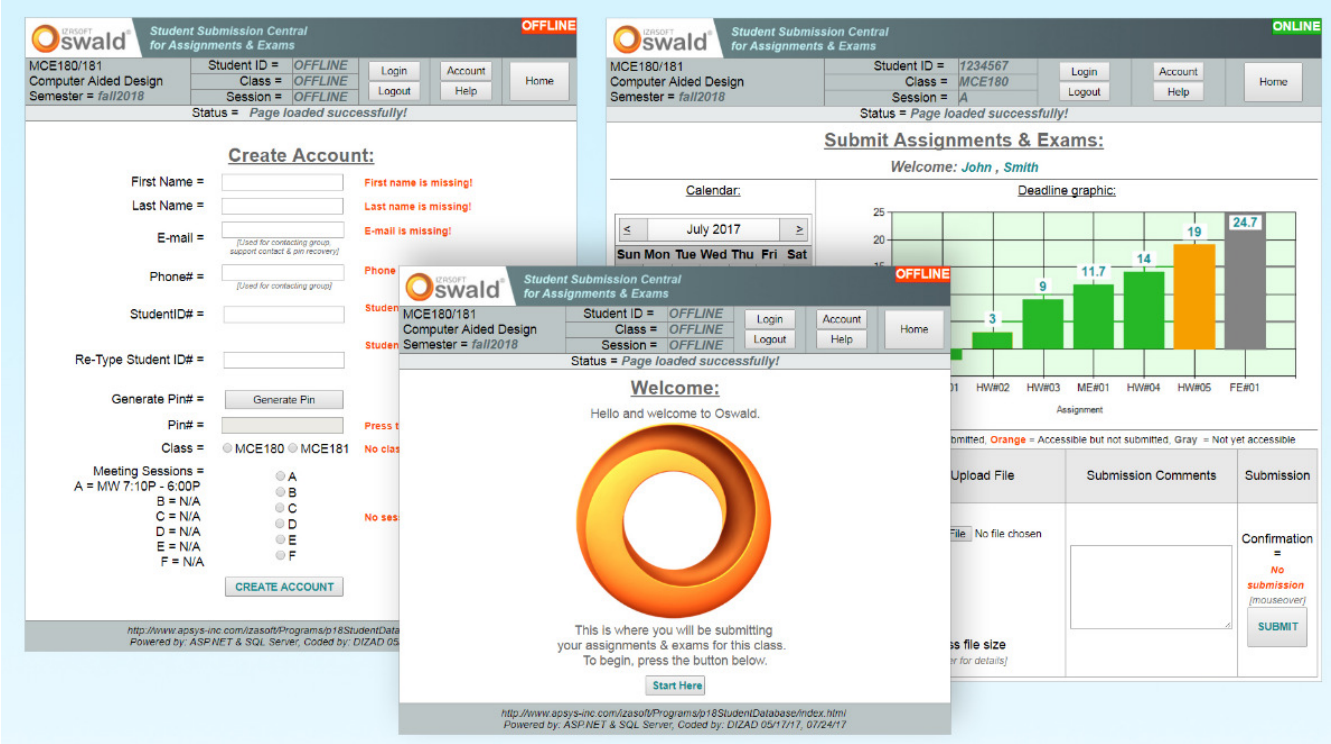

Figure01: Oswald application showing the home, create account, and submission metric screen.

1. Abstract:

This paper introduces *Oswald*, a centralized enterprise application for teaching university courses. *Oswald* improves quality of education, pedagogy, and reduces educational costs [1]. Online teaching tools in computer science are growing these days. Websites like the *Khan Academy*, *Udacity*, and *Udemy* are emerging quickly. *Oswald*'s niche is that it encompasses an all-in-one solution by using 2 modules, an instructional module and a submission module. The instructional module handles a series of progressive checkpoints that are designed to be easy to follow by the common student. The submission module handles all of the student's submissions for control and grading. Technical details on this proposed solution is found under *section 5: Architecture of the Proposed System*.

A beta version of *Oswald* was released for 1 semester (Fall 2017) at *Cleveland State University*. To date, it managed 2 engineering courses and 250 students in total. The beta version was only 25% complete, but still reduced FTEs (Full Time equivalent i.e. administrative hours) by 15% for one semester. This is equivalent to $4k of savings per year for 250 students taking a 4-credit hour class for 1 year (2 semesters). The remaining 75% of *Oswald* is still under development, but is estimated to save FTEs up to 85% (equivalent to $20k savings/year proportional to the same conditions as the 2 classes *Oswald* was tested on). The savings can expand further if the application expands to other courses, departments, and universities.

2. Introduction:

Educational online applications are growing in computer science[4]. Websites like the *Khan Academy*, *Udacity*, *freeCodeCamps*, *Microsoft Virtual Academy*, *Coursea*, *Udemy*, and the *codeAcademy* have become more and more popular in the last 5 years. *Figure02* shows a chart generated with *Google Trends* to show the increasing popularity of these online academies.

These tools have grown because of the industrial demand for software skills, and from the effectiveness of these tools in teaching computer science. This proposal will discuss an all-in-one solution, named *Oswald*, that capitalizes on the successes of these existing solutions, leaving behind their failures, and providing additional functionality to improve the learning experience for the student.

One of the common mistakes that scientists make when writing proposals, is that they sometimes fail to steer their research to ideas that are truly in demand. Regardless of prestige or technical novelty, the value of a proposal (from the stakeholder's perspective) is often measured by the amount of realistic impact it can deliver. A proposal that delivers evidence of real cost savings, quality improvements, and true improvements to people's lives (on a mass scale), will almost always carry weight amongst investors. *Oswald* plans to make an impact by using technology to tackle the following 2 problems in education: (1) improving pedagogy and (2) reducing educational costs. *Oswald* is designed to be fully-autonomous, meaning it can work on its own and manage itself without human interference.

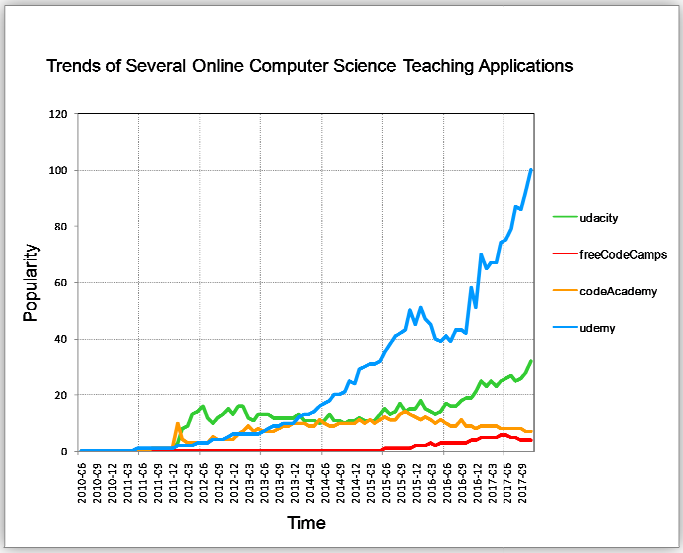

Figure02: Market analysis using Google Trends.

3. Problem Description:

3.1. Demand:

The need for an application like *Oswald* starts with the demand. There is currently a growing shortage of software engineers in the United States, where 8-10 software positions are available for every qualified candidate [15] [11]. According to the *Bureau of Labor Statistics*, software positions are expected to grow by 24% in the next 10 years [10], which is much larger than the average growth of other professions. Unfortunately, there aren't enough computer science graduates to meet this demand [12]. Completing a 4-year degree is an investment that many middle-aged Americans aren't willing to take. This is why alternative and cheaper solutions have grown over the last 5 years. Computer science programs that take less time to complete, and can be completed anywhere in the world, are becoming attractive ideas. The challenge now becomes the effectiveness of these educational tools. Can a program that lasts 2 years truly be as effective as a program that lasts 4 years?

3.2. Pedagogy in STEM:

One of the re-occurring complaints about college courses in the "hard-sciences" or STEM-related disciplines (Science, Technology, Engineering, and Mathematics) is the lack of effective pedagogy [3][13]. This trend seems to worsen as the prestige of the institute increases (elementary school -> high school -> community college -> university…etc.) [3]. Details on this issue are described below.

3.2.1. Differential Instruction:

Differential instruction is a technique used in education where a course is designed in such a way that students learn efficiently, where lessons are geared towards a student's method of learning [18]. Everyone learns differently, and to effectively teach a subject, the topic must be designed in a way that the students can understand it [18].

Teaching STEM courses is not easy for the professor. Math and science concepts are sometimes difficult to comprehend naturally as opposed to some of the softer sciences, making it challenging for a professor to teach effectively. Becoming a computer scientist is a process that requires time, effort, and dedication. Training freshmen students to this mindset is one of the challenges to effective pedagogy.

Another potential issue is that a lot of talented STEM professors could be leaving the classroom for monetary reasons. STEM-related professions pay tens of thousands of dollars more than STEM-related teaching professions [17]. This can cause a shortage of STEM professors, where universities often have to search internationally to fill these roles. This is an effective solution for maintaining credentials, but could introduce a cultural gap between the students and professor. There could be a mismatch between the learning process of the student and the learning process of the professor, affecting the idea of differential instruction [13]. For example, some cultures favor the idea of asking questions in class, because it fosters engagement. In other cultures, asking questions is seen as disruptive and subject to

ridicule, because it shows lack of understanding [13] [23]. Language barriers can also be an issue. For instance, the *University of Pittsburgh* offered $264 in tuition refunds to 9 students who could not understand their teacher in a STEM algebra class [13].

Pedagogy can also be affected if the primary focus in recruiting is research instead of teaching. For instance, in contrary to high school or community colleges, research institutes (such as universities), are more interested in research credentials than teaching abilities, because research is a big portion of a university's business. As Arthur P. Mattuck (mathematics professor at MIT) would say, "*Research comes first, then we look for some evidence that they [the students] aren't hopeless in the classroom*" [13]. Likewise, professors who were hired for their academic credentials, instead of their industrial experience, could design their courses that are geared towards academics, instead of the industry (where most of the demand exists and where most of the students end up).

3.2.2. Lack of Auditing:

There's a level of risk with the quality of a product, when it's led by a small body of people. A classroom is often led by a single professor and a textbook. This works well in a class where the instructor is knowledgeable and experienced in the topic, but can pose a risk in efficiency if the professor is teaching the course for the first time with little background on the information.

3.2.3. Holes in Pedagogy:

A hole in pedagogy is a missing piece of information (a node) that is required to fully understand the "tree" of an idea. For instance, if the instructor taught that "a = b" and "b = c", and then asked if "a = c" on an exam, then this pedagogy has no holes. Students know exactly what they need to know to answer the question. Similar to a computer program that executes properly when there are no missing lines of code in the sequence.

In reality, it's very difficult for professors to fill every hole in their pedagogy. Professors often ask whether "a = c" before mentioning that "a = b" or "b = c". It's possible for the student to discover that "a = c", but the student now has to triple their study time to discover this answer. This defeats the purpose of school, because the students can then just learn the subject on their own. The reason students take a course is to expedite the process of learning so that they can immediately use the knowledge when they are working. A research class or an independent study is designed differently, where students are trained to research problems that haven't been solved before.

The goal is to minimize the number of pedagogical holes, so that information is transferred efficiently and the economy moves forward. A data scientist in NYU identified these holes as *"long lectures with minimal examples, relying heavily on personal research and the internet for explanations, and assignments with little correlation to the industry"* [3]. Minimizing these holes is a challenge for every professor. To achieve this level of perfection, the system should rely on computers, instead of relying on human ability.

3.2.4. Knowledge Retention:

Even when all the holes are eliminated, there is still the risk of students failing to follow instructions. When a class policy is changed, it's very rare that 100% of the students will be aware of it. Class announcements and e-mails may improve the awareness, but it's still rare that the update is 100% communicated. People daydream, have attention deficiency disorders, and are rarely attentive at all times. This inefficiency results in additional wasted hours, because the instructor and teaching staff now have to spend additional time to repeat themselves.

3.2.5. Cost:

3.2.5.1. Grading:

Every industry (including research institutes) is looking for ways of minimizing costs. Money is nothing more than human time. In manufacturing, one of the most common costs (or inefficiencies) comes from the manual labor of producing parts. These costs are reduced through the use of automation technology, because automation reduces human time. In education, one of the biggest costs (i.e. human time) is in grading. Grading is a low-bandwidth procedure, and can be automated with a computer instead of a human. A university can spend as much as $150 per student per class per semester on grading. The grading can also be subjective; it depends on the teaching assistant, the mood he/she is in, and the accuracy of their deductions. By using technology to automatically grade, not only will the teaching staff save time, but the quality becomes more accurate and consistent.

3.2.5.2. Book-Keeping:

There are wasted costs in book-keeping when managing a class. The teaching staff has to keep a record of grading, grouping, or student account information. The book-keeping becomes more tedious when changes occur in the middle of the semester. Students have emergencies, exam accommodations, their assignments get lost, or data entry errors occur when cross-referencing spreadsheets. All of these errors can be reduced by using an "enterprise"-style application that can manage its own databases and "poke yoke" the errors from the manual process.

3.2.5.3. Office Hours:

There are wasted costs associated with office hours. Office hours are the result of inefficiencies in the professor's pedagogy and the student's learning methodology. A professor has to spend extra time repeating themselves, when that time could have been used more effectively doing research.

3.2.6. Problem Description Conclusion:

By improving pedagogy and reducing the inefficiencies of a research institute, tuition costs can decrease, professor salaries can increase, and the quality of education can increase.

4. Survey on Related Work:

Several teaching solutions currently exist in the market. Some solutions do a nice job specializing in the inefficiencies discussed in *section#03*, but they have trouble being an all-in-one solution.

4.1. Udacity:

Udacity is a company that partnered with universities like *Georgia Tech University* to offer online degrees in computer science for a fraction of the in-class tuition cost. To complete an online master's degree in computer science from Georgia Tech costs $6,000 and 2 years. *Udacity* provides an interface with videos and interactive exercises to teach material to students. The videos are high quality, and are created with highly qualified professionals. However, one of the downsides is that *Udacity* still relies heavily on manual labor to grade assignments and projects. The book-keeping for each student is managed manually by the professor. This is again failing to meet the all-in-one fully autonomous solution that *Oswald* is suggesting.

4.2. Khan Academy:

The Khan Academy earned a lot financial support in 2006 from Bill Gates, and has now grown to be a leading provider of online education [19]. The Khan has one of the best interfaces for learning. It's interactive, responsive, easy to use, and has a powerful artificial intelligence (built by John Resig, creator of JQuery), that guides you to the correct answer. It can detect student's errors and provide relevant feedback. The *Khan Academy* is a good teaching tool, but it doesn't have a module for student grading and submissions. Many of their courses also aren't scaled to a 16-week college course. They are mainly a series of mini-modules designed to be completed in a few days.

4.3. Blackboard / Turn-it-in:

Blackboard is an application associated with many university applications to assist in student submissions and class content. It does a good job of providing an interface that is customizable. It can store course documents like *PowerPoint* slides or *Word* documents. It can host forums where students can converse with each other online through discussion boards. It can deploy online exams that are timed and restricted. Finally, it can manage assignment submissions. *Blackboard* has a solid infrastructure, but its functions are very basic, it doesn't have a lot of artificial intelligence. *Blackboard* can't automatically grade student submissions, and store them in a student database. It also doesn't include instructional modules.

4.4. FreeCodeCamps / CodeAcademy:

The *FreeCodeCamp* and the *Code Academy* are almost fully-autonomous, and provide certificates for students who complete all of the required modules. The *Code Academy* provides certificates for learning different languages, whereas the *FreeCodeCamp* provides 3 certificates (frontend, data visualization, and backend). The student has to complete algorithms, challenges, and projects to receive the certificate. The grading is controlled through hundreds of test outputs per exercise, which run through the student's code, and algorithms grade accordingly. Students are unable to progress through the checkpoints unless their code passes these test conditions.

These academies are not only fully-autonomous, but also have legitimate credentials in the industry. Recipients of these certificates have received jobs at

IBM and Microsoft [24]. The *freeCodeCamp* and *codeAcademy* are the most similar to *Oswald.* However, similar to the *Khan Academy*, they aren't scaled to a 16-week college class.

5. Architecture of Proposed Solution:

5.1. Design Overview:

5.1.1. MVC Model:

To begin, *Oswald* runs on an MVC (Model View Controller) model using ASP.net. The data model is managed through *SQL server*, the front-end view is managed through *html / CSS / Javascript* and the back-end (Controller) is managed through *C#*. All 3 units communicate together to manage the application.

5.1.2. Server:

The server contains the correct version of *ASP.net* and *SQL server* platforms for running *Oswald*. The database is pre-populated with SQL queries and pre-made .mdf files. A connection string for this database is attached to the configurations file of the website application.

5.1.3. Centralization:

College courses are the same worldwide. *Biology* is the same regardless of university. This allows *Oswald* to stay centralized, making it easier for revision control, and allow updates to occur in one location. It also allows the course packages to be engineered to perfection. Each course package can be reviewed by teams of experts, and tested with every kind of student. Universities only need to be access nodes, which communicate with the centralized packages of *Oswald*.

5.1.4. Compression:

Several archives will be designed using SQL databases and storage service systems (similar to *Google Drive*). The SQL servers will manage the student profiles with profile data such as progression, studentID#, grades, name,…etc. The storage service system will contain the student submission files (i.e. pdf reports, dxf drawings, .c source files,…etc.). The system would extract these files for grading so that their grades can be processed inside the SQL databases. With the intent of keeping the application centralized, and expanding to universities worldwide, the data can easily grow and become expensive.

JPEG images are one of the most common types of submission files for *Oswald*. Essays, papers, math problems, and some technical drawings will all be submitted as JPEG files for analysis and automatic-grading. This is why compressing these files on a large scale will be crucial for *Oswald*'s success.

In the technical paper, "*The Design, Implementation, and Deployment of a System to Transparently Compress Hundreds of Petabytes of Image Files for a File-Storage Service*" [7], a compression tool called Lepton is introduced. Lepton is designed to compress JPEG files in a file storage service system. Through the JPEG compression pipeline of color transformation, down-sampling, forward discrete Fourier transform, quantization, and encoding, Lepton proposes significant efficiency improvements to the encoding stage. It achieves this by using a probability model and arithmetic coding (instead of Huffman coding). The tool can compress JPEG images by 23% on average and ran smoothly for one year under the Dropbox file storage service system.

5.2. Instruction Module Design:

5.2.1. Introduction:

The instruction module is a set of progressive exercises/checkpoints that the student goes through to complete the course. It contains a series of short videos and exercises that are designed to be easy to follow. The module uses algorithms to recommend solutions when students are stuck and a Q & A module if further assistance is needed.

5.2.2. Subject Matter Experts:

One of the key features of *Oswald* is that it is designed by SMEs (Subject Matter Experts) not only on the technical side but also in teaching. The technical experts not only come from the industry, but are some of the experts who contributed to the concepts discussed in a technical textbook. In computer science, (for when some were alive) this could include Edsger Dijkstra, James Gosling, or Dennis Ritchie. These educators are the founders of fundamental concepts in computer science, where they've tested, implemented, and truly understood the importance of the concepts they published. Their input would improve tomorrow's education because the knowledge is coming from the source, as opposed to a trickled-down textbook chapter that a professor tries to decipher.

Once the technical experts finished their part, the teaching experts would then take their work and

design *Oswald* so that the lessons are easy to follow. They would reduce pedagogical holes, design lessons that are incremental, and test frequently with different students, and revise as needed.

5.2.3. Recommendation Algorithms:

One of the important features of *Oswald* is its ability to recommend solutions when students are stuck. Recommendation algorithms can pick up these queues from the student's feedback, and provide information for them so that students can complete their next step.

Amazon has done a lot of research in recommendation algorithms, and published a paper about item-to-item collaborative filtering [9]. The publication talked about methods of "estimating" recommendations for a user, based on other customers (historical purchases / customer clusters), and products (attributes such as authors and genres). The existing methods had issues with scaling to large data, speed, and quality of recommendations. However, *Amazon* found out that by using item-to-item filtering (where recommendations are based on other items that customers purchased on similar orders), they were able to increase speed, scale to large data, and improve the recommendation quality [9].

The paper discussed recommendations for products inside the Amazon catalog. However, similar algorithms can be used for "estimating" where students (or professionals that are debugging programs) are stuck when they go through the *Oswald* exercises. The recommendation algorithm can work several ways. First, similar to the item-to-item filtering algorithm, the algorithm can assess an archive of other students who made similar mistakes, and recommend a solution based on their probability of success. Second, based on the "Differential Instruction" concept discussed in *section#03.02.01*, the cluster model would be effective, by recommending solutions geared towards clusters of students who learn in similar ways. The algorithm can use a series of supervised machine learning recommendations geared towards each cluster of student.

5.2.4. Question & Answer With AI:

When a recommendation isn't enough, a Q & A sub-module becomes relevant. *Oswald* needs a way to take a question and reply to it quickly and accurately. This sub-module is critical in keeping *Oswald* autonomous, so that *Oswald* can operate at high volume and save costs.

One of the most significant studies in this area is the *IBM Watson* operator under the paper "*Building Watson: An Overview of the DeepQA Project*" [6]. The paper discusses a revolutionary artificial intelligence tool called *Watson*; that specializes in answering questions. Analogous to *Deep Blue* for chess, *Watson* was designed for Q & A on the game show *Jeopardy*. *Watson* behaves similar to the way the human brain determines answers, by selecting the response with the highest probability of success. *Watson* calculates these probabilities based on a series of sources / methods such as process of elimination, reliability-checking (i.e. avoiding unreliable sources such as *"the Onion"* or *"The Huffington Post"* for answers), and adapting to historical mistakes. *Watson* searches for information from a series of archives. It relies on the internet and its own internal database (for speed optimization). In 2011, Watson competed against former *Jeopardy* champions and won 1st place and $1million [21].

Watson can be used by *Oswald* for answering student questions. *IBM Watson*, and other similar applications like *IBM Deep Blue* and *Amazon Alexa*, have a record of success in beating humans, and have a lot of potential for other applications outside of chess, jeopardy, or medical diagnosis. The mixed architecture of machine learning, artificial intelligence, and searching through multiple archives is a solid structure for the instructional module of *Oswald*. The feedback is accurate and instant, where the student doesn't have to wait for a response from an e-mail.

5.2.5. Making Students Want to Learn:

One of the ways of attracting students in computer science is by keeping the exercises entertaining, by turning *Oswald* into a video game. Video games are entertaining because they deliver a high rate of accomplishment and players level-up as they progress. For instance, as the student progresses through the lessons, the student can unlock perks, such as extra time on exams, if the student turned in assignments on time throughout the semester. Another idea can be keeping an interactive scoreboard, showing the student's ranking amongst an archive of other students who took the course. Scoreboards could incline a competitive atmosphere and encourage students to study harder.

The assignments can also be more exciting, like solving real problems from the industry. One of *Elon Musk*'s biggest complaints about education was that tools are taught without teaching the application. To resolve this, he built a school were students are presented with real problems, and the students learn how the tools are used to fix the problems. Therefore, the students learn the tools automatically, without forcing themselves. His methods were so effective, that Elon says the students say "*the vacations are too long*" [25].

The idea of keeping education like a video game was partially implemented in *Oswald Beta,* by using a graphic showing the student's deadlines as the semester progresses (see *figure03*). After comparing late penalties to previous semesters, this graphic psychologically inclined students to submit their assignments on time by 26.2%, compared to pre-*Oswald* semesters.

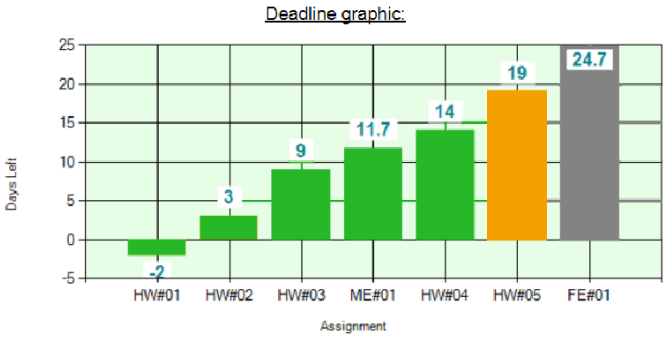

Figure03: Oswald submission control.

5.3. Submission Module Design:

5.3.1. Assignment Selection:

In order to keep every assignment unique, *Oswald* generates exams and homework assignments by randomly selecting questions from an archive. As soon as a student creates their account, every exercise, assignment, and exam is pre-determined for that semester.

5.3.2. Submission Control:

For some classes, submission control becomes very relevant. The submission control that *Oswald* introduces is much better than the one that *Blackboard* offers. For instance, in solid modeling, it's crucial for student submissions to be a certain file type and size. If the student fails to submit the right file, the teaching staff won't be able to grade their files, because external references can be broken, and the submission would be a blank sheet of paper on the screen to grade. This can cause students to lose significant points for an improper submission, as opposed to losing points for miss-understanding a core concept of the class. Resolving these issues have led to many wasted hours.

Oswald helps mitigate this by using a series of ASP.net validators, as shown in *figure04*, to control for file type, size, and presence. This sort of implementation is called "poke-yoke" or "error-proofing" in manufacturing, and is the key to zero customer disappointment (ZCD). Students are no longer blindly submitting e-mails, they are submitting files in an interface that provides instant feedback, and will notify students immediately, if their submission is valid.

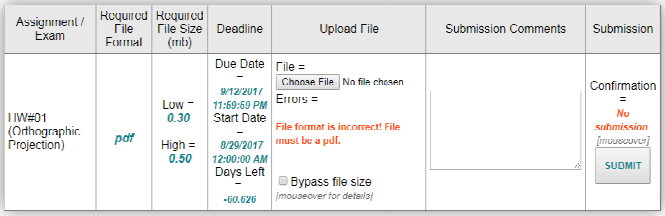

Figure04: Oswald submission control with red indicators.

When students submit files, *Oswald* will automatically generate a clean e-mail for the teaching assistants to grade. *Oswald* will extract the student's profile information like name and studentID#, and attach the submitted file in the email. This significantly reduced human errors from previous semesters, where students would often misspell their names, misspell their studentID#s, or forget to attach files when sending e-mails to the teaching assistants. Although book-keeping errors may seem insignificant in most mid to small size classes, tools like these are crucial for classes that have over 250 students.

5.3.3. Auto-Grading:

5.3.3.1. Multiple-Choice:

One type of test method is using multiple choice. Even thought multiple choice is one of the easiest methods of grading automatically, many professors still spend time to grade each question manually. *Scantron* technology is also not much of an improvement because the professor has to spend time running the scans, cross-referencing spreadsheets, and manually entering the scores on the computer. *Oswald* will store student answers inside a series of normalized tables, and compare them with the answer

table. The weights are adjusted, the awarded points are calculated, and the awarded points are stored inside *Oswald*'s relational tables.

5.3.3.2. Data-Entry:

Another type of submission is using numerical input through a textbox. This can also be easily graded, especially with partial credit. The number can be compared to the answer, and deduct points proportional to the offset. This technique is used in many standardized certificate exams. For instance, in the *Certified SolidWorks Professional Exam*, participants are asked to model geometries where the volume of the part (in in^3) is the final answer [20]. Instead of using complex object recognition, the application evaluates a single volume number, instead of the geometric accuracy.

5.3.3.3. Drawing Recognition:

Another type of submission is a technical drawing. These require a little more complexity to evaluate, because the type and location of each line needs to be compared with a solution drawing. The student's lines are stored in a series of linked lists, where each line type is a different structure. As the lines are drawn, the linked list is incremented with newly added x and y coordinates. When lines are removed, the linked list finds the closest coordinate to the mouse cursor, deletes those coordinates, and stitches the previous and next point of the line. When the drawing is submitted for evaluation, the linked lists are serialized into a multi-dimensional structure that can be de-serialized for grading. An *Oswald* work-in-progress prototype is shown at *izasoft.somee.com* [22].

5.3.3.4. Computer Vision:

The most complex type of submission to grade is an image. The image can be an essay or math problem that requires object recognition to evaluate. The paper "*Pictorial Structures for Object Recognition*" suggests an efficient framework for modeling and recognizing objects in an image [8]. The framework generates a series of deformable parts that are arranged in a way that represents the unrecognizable object. Each part is modeled separately and deformable using spring-like connections between the parts. The model is suitable for generic recognition problems and allows for qualitative definitions of visual appearances. The framework was tested on limb and facial recognition applications and delivered successful results.

Object recognition will be an essential solution for grading images in *Oswald*. The solution will need efficient object recognition algorithms for calculating the "closeness" of the student's work to the solution structure. Proper offset factors will need to be recognized so that unrelated mistakes won't deduct significant points from the student's grade.

5.3.4. Instant Student Feedback:

Another advantage of *Oswald* is the instant feedback that it provides. Students no longer have to wait to receive their grades; they can check it at any time by logging in to their account. The grade sheet can also automatically calculate a class curve, by instantly assessing the overall grade of the class.

5.3.5. Auto-Grouping:

Another time-consuming task in managing courses is the process of grouping students for project assignments. Whether the grouping is randomized, student selected, or clustered on grade performance, grouping methods have mainly been manual for the most part. The teaching staff has to manually assess student feedback, find their account in cross-functional charts, and determine a group number accordingly. This becomes challenging when group numbers are changed in the middle of the semester (for various emergency reasons). In classes where it is critical for group quantities to stay the same(for budgetary or logistical reasons of the class), these mid-semester changes can cause an imbalance in the group sheets, and the teaching staff wastes more time updating the archives. When a member leaves a group, it leaves a hole in their spot. This can be difficult to re-arrange, especially in a class where (for example) groups of 3 are required and multiple groups of 2s are left in the system. The teaching staff now has to communicate with the members that are willing to split (even if the students have already progressed through the project) so that groups of 3 are fulfilled.

To resolve these issues, *Oswald* introduces grouping algorithms that instantly communicates the status of a student's group. In *Oswald*, every student has a built-in profile with their student information, meaning cross-referencing with external campusnet spreadsheets and grade sheets are no longer needed. *Oswald* can communicate instantly to any of its databases.

Next, a module is incorporated where students can either select their own groups, randomize, or cluster

based on grade performance. *Oswald* matches the provided student ID# with other students that don't already have a group.

Oswald can also incorporate deadlines when setting groups. For instance, the instructor may want students to pick their groups by a certain time, and have the system randomize the remaining groups after a certain date. *Oswald* can achieve this using events that trigger randomization functions after the deadline *DateTime* variable has passed the *DateTime* of now. This implementation is shown in *figure05*.

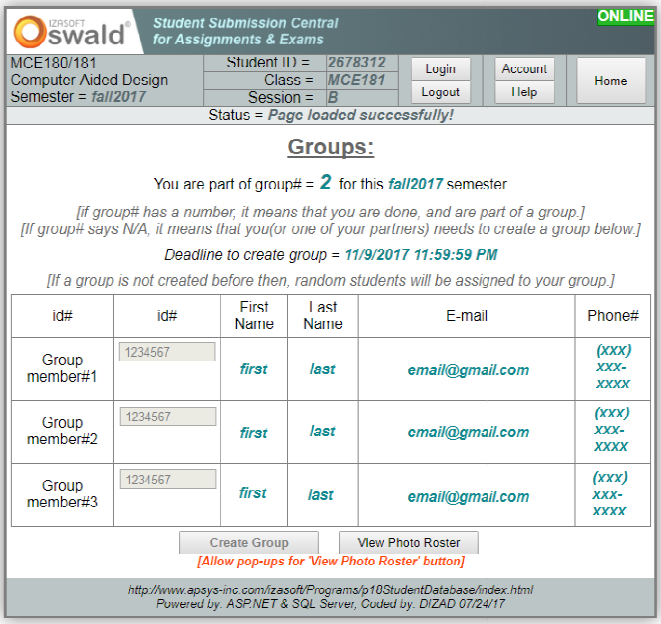

Figure05: Automatic grouping.

6. Performance:

Oswald Beta performed well on its pilot semester. As shown in the combo bar chart of *figure06.* The application reduced teaching time per student by 15.4% for one semester, and maintained equivalent quality.

The average time spent was based on the collective logged administrative hours of *Daniel Izadnegahdar* (professor), *Vivek Reddy Katukuri* (teaching assistant#01), and *Ebraam Kamel* (teaching assistant#02). The hours consist of answering student questions, grading, and book-keeping. The data was compared with previous semesters when *Oswald* was not used.

The quality rating was measured from student feedback through a series of survey tools including campusnet, and the netSupport podium survey application.

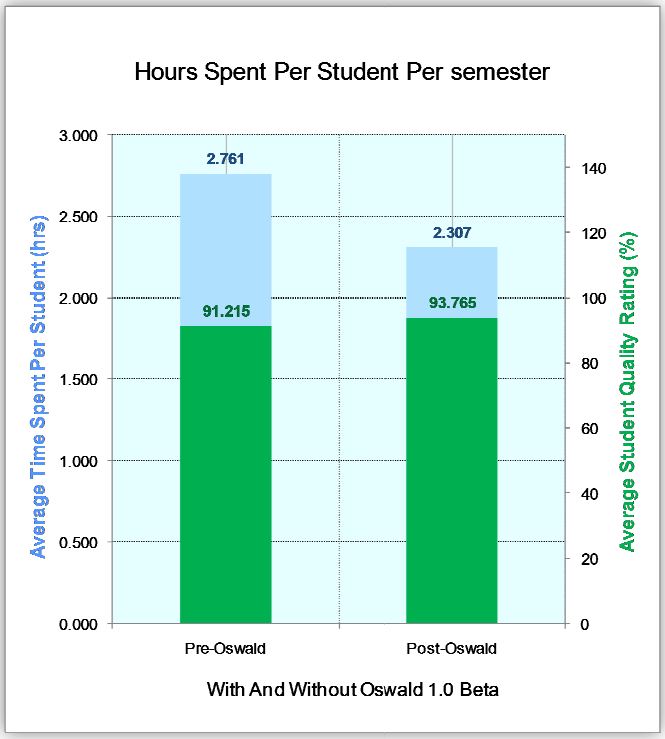

Figure06: Oswald performance comparison.

7. Historical Successes:

Oswald was developed by *Izasoft*, a consulting company specializing in automation [5]. *Izasoft* consists of engineers, professors, and technical consultants. *Izasoft* built a brand of delivering reliable solutions in pedagogy, mechanical automation, and software automation. Its goal is to continue this trend with the release of *Oswald*.

Izasoft had historical successes in effective pedagogy. In 2015, it published a book on engineering design using SolidWorks[14]. The book received positive reviews, where several students mentioned that the book was easier to follow than the alternative books that were published in that field.

Izasoft also had successes in automation. It designed and implemented a series of systems for several manufacturing companies in northeast Ohio. In total, these machines are saving $15k - $20k per month.

8. Lessons learned:

Oswald Beta performed well overall, but lessons were learned during the testing phase. The 1st error was in

determining current time. There are many existing methods for determining current time, such as the *dateTime.Now()* function. However, when working with servers at multiple locations, this function can deliver incorrect information. For example, during localized testing in Ohio, the *dateTime.Now()* function returns Ohio time correctly. But when the MVC application is deployed in a server in California, the *dateTime.Now()* function may return a time with a 3 hour offset. If control labels aren't displaying this data, this error can be unnoticed. A better design decision was to use the universal time clock method, with a correction factor for the time zone and daylight saving time. By using this technique, the application guarantees to deliver the time of the desired area, no matter where the server is located.

Another lesson learned dealt with the reliability of the server. During certain traffic cycles, the server would randomly go offline and online. This was a big issue during deadline periods of the semester, because students couldn't submit their assignments. Testing for reliable server performance was a key lesson of *Oswald Beta.*

9. Proposed Approach / Plan of the Project:

The plan is to continue to test the beta version of *Oswald* for several semesters. Once this is achieved, *Oswald 2.0* will be tested, including most of the grading module complete. The instructional module currently exists through a series of .pdf files, but will eventually be integrated into *Oswald 3.0*, as an integrated interactive interface similar to the *freeCodeCamps*.

10. Extended Future Plan:

10.1. Expanding the Curriculum:

To date, *Oswald* only managed 2 college courses, but the future plan is to manage an entire curriculum of courses, not only in STEM, but in all other disciplines. If students trust the *Oswald*-brand, as a product that delivers courses that are easy to learn, than *Oswald* can expand to an entire catalog of university courses.

10.2. Expanding to Other Industries:

Oswald can also expand to the industry for helping error-proof engineering and computer science files. The auto-grading algorithms of *Oswald* can be used in the industry for providing feedback on areas where their engineering documents may be prone for errors.

10.3. Open-Source Editing:

Another future plan is to keep *Oswald* open-source, where users can edit it and fill in any pedagogical hole that they find. *Oswald* would be a self-adapting application that improves itself, based on the edits that users make, similar to *Wikipedia*. This addition may require a team of moderators, to assure the changes are appropriate and adding value to the application.

10.4. Security:

With a centralized auto-grading application, security becomes an issue. If *Oswald* is not secure, students can change their grades to whatever they want. A talented hacker can also delete segments of other student's profiles. The fact that *Oswald* is centralized is a benefit for revision control, but challenging for security. The next phase of *Oswald* will incorporate proper encryption algorithms to assure that student profiles are protected.

11. Conclusion:

With the growing demand of software skills, effective pedagogy in computer science is needed. Existing applications are growing, but are failing to provide an all-in-one solution that a research institute can integrate. *Oswald* is a proposed all-in-one solution that tackles 2 main problems in education: (1) effective pedagogy and (2) reducing educational costs. It resolves these issues by using recommendation algorithms, artificial intelligence, database management, and an alternative architecture design. *Oswald* ran for 1 semester, and saved 15%, or $4k in administrative FTEs. The goal is to continue developing *Oswald*, until all of its functionalities are complete and tested. Once this is done, it plans to expand to additional courses and universities.

9. References:

[1] *Oswald application, Oswald.somee.com*

[2] *Khan Academy website, https://en.wikipedia.org/wiki/Khan_Academy*

[3] *The Missing Pedagogy in Computer Science. https://blog.datapolitan.com/2014/09/02/the-missing-pedagogy-in-computer-science/*

[4] *Educational Online Tool Google Trends https://trends.google.com/trends/explore?date=all&q=Udacity,FreeCodeCamp,CodeAcademy,Udemy*

[5] *The Izasoft Company, http://www.apsys-inc.com/izasoft/*

[6] *Ferrucci David, Brown Eric, Chu-Carroll Jennifer, Fan James, Gondek David, Kalyanpur Aditya, Lally Adam, Murdock William, Nyberg Eric, Prager John, Schlaefer Nico, Welty Schlaefer, Welty Chris. "Building Watson: An Overview of the DeepQA Project", Artificial Intelligence Magazine, September 2010, pp59-79.*

[7] *Horn Reiter Daniel, Elkabany ken, Lesniewski-Lass Chris, "The Design, Implementation, and Deployment of a System to Transparently Compress Hundreds of Petabytes of Image Files for a File-Storage Service", 14th USENIX Symposium on Networked Systems Design and Implementation, March 2017, pp01-17.*

[8] *Felzenszwalb F. Pedro(MIT), Huttenlocher Daniel(Cornell), "Pictorial Structures for Object Recognition", International Journal of Computer Vision, January 2015, pp01-42.*

[9] *Linden Greg, Smith Brent, York Jeremy, "Amazon.com Recommendations Item-to-Item Collaborative Filtering", IEEE Internet Computing, IEEE Computer Society, February 2003, pp76-80.*

[10] *Bureau of Labor Statistics, https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm*

[11] *freeCodeCamp, the Official Website https://www.freecodecamp.org/*

[12] *Calling All Computer Science Majors: Jobs are Waiting for You http://college.usatoday.com/2017/02/15/calling-all-computer-science-majors-jobs-are-waiting-for-you/*

[13] *Education and Communication, https://goo.gl/UhBSmA*

[14] *Engineering Design Using SolidWork, https://drive.google.com/open?id=1KA-nKllqokIIYXvpBASNYwYabK5cYykS.*

[15] *You Probably Should Have Majored in Computer Science, https://qz.com/929275/you-probably-should-have-majored-in-computer-science/*

[16] *Professors and Recruiting, goo.gl/phQnEA*

[17] *Is Stem Education in Permanent Crisis? https://www.edweek.org/ew/articles/2016/10/26/is-stem-education-in-permanent-crisis.html*

[18] *Differential Instruction, https://en.wikipedia.org/wiki/Differentiated_instruction*

[19] *Khan Academy, https://en.wikipedia.org/wiki/Khan_Academy*

[20] *Certified SolidWorks Professional Exam, http://www.solidworks.com/sw/support/797_ENU_HTML.htm*

[21] *IBM and 'Jeopardy!' Relive History With Encore Presentation of 'Jeopardy': the IBM Challenge, https://web.archive.org/web/20130616092431/http://www.jeopardy.com/news/watson1x7ap4.php*

[22] *Technical Drawing Evaluator, http://www.izasoft.somee.com/*

[23] *American Culture & Values, http://www.internationalstudentguidetotheusa.com/articles/culture.htm*

[24] *FreeCodeCamp Industrial Credentials https://www.linkedin.com/school/4831032/*

[25] *How Elon Musk is Educating His Children https://qz.com/414326/how-elon-musk-is-educating-his-children/*