

Grants Collection

Kennesaw State University



UNIVERSITY SYSTEM
OF GEORGIA

Rebecca Rutherford, Dawn Tatum, Susan VandeVen, Richard Halstead-Nussloch, James Rutherford, and Zhigang Li

Discrete Structures



Grants Collection

Affordable Learning Georgia Grants Collections are intended to provide faculty with the frameworks to quickly implement or revise the same materials as a Textbook Transformation Grants team, along with the aims and lessons learned from project teams during the implementation process.

Each collection contains the following materials:

- **Linked Syllabus**
 - The syllabus should provide the framework for both direct implementation of the grant team's selected and created materials and the adaptation/transformation of these materials.
- **Initial Proposal**
 - The initial proposal describes the grant project's aims in detail.
- **Final Report**
 - The final report describes the outcomes of the project and any lessons learned.



Unless otherwise indicated, all Grants Collection materials are licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Initial Proposal

Application Details

Manage Application: Textbook Transformation Grants: Round Eleven

Award Cycle: Round 11

Internal Submission Deadline: Tuesday, January 23, 2018

Application Title: 354

Application ID: 002074

Submitter First Name: Rebecca

Submitter Last Name: Rutherford

Submitter Title: Department Chair, Professor of IT

Submitter Email Address: brutherf@kennesaw.edu

Submitter Phone Number: 470-578-7399

Submitter Campus Role: Proposal Investigator (Primary or additional)

Applicant First Name: Rebecca

Applicant Last Name: Rutherford

Applicant Email Address: brutherf@kennesaw.edu

Applicant Phone Number: 470-578-7399

Primary Appointment Title: Department Chair, Professor of IT

Institution Name(s): Kennesaw State University

Co-Applicant(s): Dr. Richard Halstead-Nussloch, Prof. Dawn Tatum, Prof. Susan VandeVen, Prof. James Rutherford, Zhigang Li

Submission Date: Tuesday, January 23, 2018

Proposal Title: 354

Proposal Category: No-Cost-to-Students Learning Materials

Final Semester of Instruction: Fall 2018

Are you using an OpenStax textbook?: No

Team Members (Name, Email Address):

Dr. Becky Rutherford - brutherf@kennesaw.edu

Dr. Rich Halstead-Nussloch - rhalstea@kennesaw.edu

Prof. Dawn Tatum - dtatum7@kennesaw.edu

Prof. Susan VandeVen - svandev@kennesaw.edu

Prof. Jim Rutherford - jruther3@kennesaw.edu

Dr. Zhigang Li - zli8@kennesaw.edu

Sponsor, (Name, Title, Department, Institution):

Dr. Rebecca H. Rutherford

Interim Assistant Dean of the College of Computing & software Engineering, and Department Chair, Information Technology

Information Technology Department

Kennesaw State University

Course Names, Course Numbers and Semesters Offered:

IT 3123 - Hardware/Software Concepts- every semester - 3 fall, 3 spring, 2 summer

IT 3223 - Software Acquisition & Project Management- every semester - 3 fall, 3 spring, 2 summer

IT 4683 - Management of IT & Human Computer Interaction- every semester - 2 fall, 2 spring, 2 summer

IT 4723 - IT Policy and Law- every semester, every semester - 2 fall, 2 spring, 2 summer

CSE2300 - Discrete Structures- every semester - 4 fall, 4 spring, 2 summer

List the original course materials for students (including title, whether optional or required, & cost for each item):

1. IT 3123, The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology Approach, Englander, 5th edition, John Wiley and Sons, 2014; ISBN-13:978-1-118-32263-5; required; cost: \$150.00; yearly enrollment: 225; total cost: \$33,750.2. IT 3223, a) Guide to Software Development, Springer Pub., ISBN 978-1-4471-2299-9; required; cost: \$101.20; yearly enrollment 245; total cost: \$24,794 b) Fundamentals of Project Management, 4th edition, AMACON; ISBN 978-0-8144-1748-5; required; cost: \$18.75; yearly enrollment 245; total cost: \$4593.75. Total for class cos: \$29,387.75.3. IT 4683, Using MIS 2017, Kroenke, 10th edition, ISBN 978-0-1346-0699-6; required; cost \$223.15; yearly enrollment 90; total cost: \$20,083.50.4. IT 4723, The Legal Environment of Business and Online Commerce, 8th edition, Cheeseman, Prentice-Hall, ISBN: 978-013-397-3310; cost: \$148.15; yearly enrollment 140; total cost: \$20,741.5. CSE 2300, Discrete Mathematical Structures, 6th ed, Pearson, ISBN: 978-0-13-469644-7; cost: \$94.97; yearly enrollment 425; total cost: \$40,362.25 All cost of books are prices for new books.

Average Number of Students per Course Section: 29.6

Number of Course Sections Affected by Implementation in Academic Year: 38

Average Number of Students Per Summer Semester: 216

Average Number of Students Per Fall Semester: 420

Average Number of Students Per Spring Semester: 425

Total Number of Students Affected by Implementation in Academic Year:	1125
Requested Amount of Funding:	30,000
Original per Student Cost:	\$736.22
Post-Proposal Projected Student Cost:	0
Projected Per Student Savings:	\$736.22
Projected Total Annual Student Savings:	\$144,324.50

Creation and Hosting Platforms Used ("n/a" if none):

Kennesaw State University D2L Brightspace

Project Goals:

In this project, we propose to take a department-wide effort to transform five required undergraduate Information Technology major courses using no-cost-to-students learning material. This project not only aims to reduce the financial burden imposed by high cost of textbooks, but also strives to develop free and open-access learning materials that offer equivalent or better educational effectiveness than traditional textbooks. These courses will then be sent through the campus Quality Matters rubric to meet institutional standards of excellence as the Information Technology degree can be completed face-to-face or completely online.

Goals:

1. Transform five required undergraduate IT major courses using no-cost-to-students learning materials
2. Create Quality Matters “ready” courses to meet institutional standards of excellence for face-to-face and online courses.

Statement of Transformation:

Research According to Priceonomics (<http://priceonomics.com/which-major-has-the-most-expensive-textbooks/>), an average undergraduate student annually spends \$1,200 on textbooks. In addition, out of 31 majors at the University of Virginia, Computer Science (and IT) comes in 8th for the most expensive books. On the other side, the University of Virginia reports that Computer Science (and IT) textbooks only have a 25% resale value based on the original price. The highest resale value for other majors is up to 70%. Previous ALG Grant Information

One Team members was part of the round two of an "Affordable Learning Textbook Transformation Grant" in 2015 (round two, award #119). They designed and evaluated the effectiveness of no-cost-to-students learning materials for database courses in the IT department, and saved students \$110,419. The assessment results showed that the developed free material offered equivalent or better learning experience than the textbooks did. The preliminary results of the grant were published in the Proceedings of Southern Association for Information Systems Conference (SAIS 2016), the final results were published in the Proceedings of the ACM Special Interests Group in IT Education (SIGITE 2016), "Transforming IT Education with No-Cost Learning Materials". They also hosted a panel discussion on no-cost learning material in IT education, at SIGITE in October 2016. The panel attracted a lot of attention among computing faculty. Many colleagues from different states were impressed with the USG initiative and with course material developed by the team. Building on our past success and lessons learned from the prior ALG grant, we will continue our transformation efforts by developing no-cost learning material for five required undergraduate IT courses. The Stakeholders There are two primary sets of stakeholders for this proposal – the students taking the five required IT classes (both in-class and online students), and the faculty developing and teaching those courses. The high cost of textbooks puts a large financial burden on students and may become a road-block for students' ability to finish their education. Our team of investigators strives to make higher education more affordable to the students. The information technology required courses listed for this grant proposal have resources that are publicly accessible, free, or with an open license to use. These materials include open and free tutorials, books, videos, labs, software, and services. One of the major problems with using regular textbooks for IT courses is that information technology material is constantly changing. Textbook publishing cannot keep up with these fast changes in the technology field. In addition, tools and software packages that are part of a textbook also become obsolete. As soon as a new version of a tool or software package is released, the instructions in a textbook become obsolete. Therefore, we need to include the latest available tools to prepare hands-on labs. Digital delivery of the learning materials makes it easier to keep the content up-to-date. Developing and assembling a set of learning materials for major courses is a unique approach. It will allow us to better align the learning material not only with the outcomes of each course, but also with the outcomes of the Information Technology program. Compared to traditional textbooks, the open source software and web resources have many benefits: 1) the Web resources are generally free to use; 2) they are constantly being updated and always reflect the latest trends and industrial development; and, 3) the materials from the Web are also more dynamic and interactive. The pitfalls of Web resources are that they are often disorganized and may contain inaccurate information. However, members of our team of investigators are not only subject matter experts in the information security field, but also proficient educators who on average have more than 10 years teaching experience including online teaching. We will select, organize and integrate resources from the web and transform the information into instructionally sound learning materials for the proposed courses including content that the team members develop themselves. We strongly believe that the new learning materials will offer up-to-date,

equivalent or better learning effectiveness compared to the original textbooks. Digital delivery also allows us to add interactive elements into the learning materials. The interactive content will not only engage the students, but also improve their learning experience. It will help to enhance the learning outcomes and learning satisfaction. The impact of our transformation efforts will be profound. By our estimates, more than 1125 students will benefit from the no-cost learning material each year. Moreover, it will benefit more students in the Bachelor of Science in Cybersecurity (eMajor) approved by the Board of Regents. One of the required courses proposed for this grant is also part of the BS in Cybersecurity. Student numbers are not included for the cybersecurity degree in this grant, but the expectation is that there will be an additional 120 students for this course per year within two years. The goal of eMajor is to reduce the cost of education by using prior learning assessments, lower tuition and potentially no-cost learning materials (<https://emajor.usg.edu>). The proposed project is expected to save current students \$144,324.50 in textbook costs each year (not counting the cybersecurity savings). Because of the cost savings from not having to buy textbooks, students may be able to take a few more courses each year and graduate sooner. Having a series of required IT courses adopting no-cost-to-student material not only offers better and more consistent learning experience to students, but also makes our nationally renowned IT programs more affordable. As a result, our IT programs could recruit more students and produce more qualified IT professionals that Georgia needs. Our experience gained in this transformation project could be useful to other programs or departments who want to lower the cost of education to their students in IT programs across Georgia. In summary, we believe the proposed project will have a positive impact in students' retention, progression, and graduation at program, department and institution levels. As shown in the following table, the textbooks used in the five required IT undergraduate major courses are expensive. In fact, most textbooks used in Information Technology are costly in general. In addition, due to the fast evolving nature of the technology field, the textbooks used in the proposed courses are updated frequently, which negatively impacts their resale value to the students. The goal of our transformation is to replace the textbook used in the proposed courses with no-cost-to-students learning materials that offer equal or higher educational effectiveness.

Data Table 1: Enrollments and Projected 2018 Enrollments of 5 IT courses

Course	Spring 2017	Summer 2017	Fall 2017	Total 2017	Projected 2018 Enrollment	Number of Sections	Total Number of students
IT3123	93	40	78	211	8	225	IT3223
	112	39	84	235	8	245	IT4683
	0	41	41	82	6	90	IT4723
	50	38	47	135	6	140	CSE2300
	170	58	170	398	10	425	Total
	425	216	420	1061	38	1125	

As shown in the following table, the textbooks used in the five required IT undergraduate major courses are expensive. In fact, most textbooks used in Information Technology are costly in general. In addition, due to the fast evolving nature of the technology field, the textbooks used in the proposed courses are updated frequently, which negatively impacts their resale value to the students. The goal of our transformation is to replace the textbook used in the proposed courses with no-cost-to-students learning materials that offer equal or higher educational effectiveness.

Table 2: Costs of Current Textbooks for 5 IT Courses

Course	Textbook Used	Cost per Student	Projected Enrollment	Projected Costs
IT3123	IT 3123, The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology			

Approach, Englander, 5th edition, John Wiley and Sons, 2014; ISBN-13:978-1-118-32263-5; required; \$150.00 225 \$33,750 IT3223 3223, a) Guide to Software Development, Springer Pub., ISBN 978-1-4471-2299-9; required; cost: \$101.20; total cost: \$32,384. b) Fundamentals of Project Management, 4th edition, AMACON; ISBN 978-0-8144-1748-5; required; cost: \$18.75 \$119.95 245 \$29,387.75 IT4683 IT 4683, Using MIS 2017, Kroenke, 10th edition, ISBN 978-0-1346-0699-6; required; \$223.15 90 \$20,083.50 IT4723 IT 4723, The Legal Environment of Business and Online Commerce, 8th edition, Cheeseman, Prentice-Hall, ISBN: 978-013-397-3310; cost: \$148.15; yearly enrollment 200; total cost: \$29, 630. \$148.15 140 \$20,741 CSE2300 CSE 2300, Discrete Mathematical Structures, 6th ed, Pearson, ISBN: 978-0-13-469644-7; cost: \$94.97 \$94.97 425 \$40,362.25 Total: \$736.22 1125 \$144,324.50

Transformation Action Plan:

With a coordinated effort, our team of investigators plan the following activities to transform 5 required Information Technology courses to completely use no-cost learning materials:

1. Research and identify no cost reading materials for each of the learning modules in each course. The reading list includes both required readings and optional readings. All of these readings will be publicly accessible, free to use, or openly licensed.
2. Research and identify no cost materials that can be shared across the courses.
3. Develop study guides and lecture notes for students' use to review course content and key learning points.
4. Adopt or develop content, assignments, exercises and lab materials that are no cost to students to replace the ones in the textbooks.
5. Develop test banks to replace the ones in the textbooks.
6. Adopt open source or no-cost-to-student lab ware for students to gain hands-on experience.
7. Update the syllabus to include major resources and no cost materials.
8. Re-develop the proposed courses in our learning management system, D2L Brightspace, following Quality Matters™ standards and get the course approved for online instruction.

The responsibilities of each investigator is described as follows.

Dr. Rebecca Rutherford, IT 3123, Project lead; Subject matter expert, course developer and instructor of record of IT 3123.

Prof. Susan VandeVen, IT 3223, subject matter expert, course developer and instructor of record for IT 3223.

Dr. Richard Halstead-Nussloch, IT 4683, subject matter expert, course developer and instructor of record for IT 4683.

Prof. Dawn Tatum, IT 4723, subject matter expert, course developer and instructor of record for IT 4723.

Prof. James Rutherford, CSE 2300, subject matter expert, course developer and instructor of record for CSE 2300.

Dr. Zhigang Li, Provide Instructional Design Support to all five proposed courses.

All course design with the no-cost materials will be provided through D2L Brightspace for our students and on the ALG website for the public access.

Quantitative & Qualitative Measures: The investigators plan to assess the effectiveness of our proposal in two ways. Qualitatively, we will design a survey and gather inputs from the students after they use the no-cost learning material. Quantitatively, we will compare students' performance data gathered from sections using traditional textbooks and sections using no-cost learning material. The investigators will collect student performance data such as pass rates from the five proposed courses taught with a textbook by team members for spring, summer and fall 2017. This data will be used as a baseline for comparison of student performance in courses with alternative no cost material. Our assessment plan can be summarized as follows. 1. Student performance measures. This data is from the overall class performance based on the grading of student works. Metrics include: * Class average, grades distribution, pass rate for each grading item. * Overall letter grades distribution, pass rate, withdraw rate, and fail rate. * Percentage of students meeting or exceeding learning outcomes 2. Specific survey on no-cost learning materials. A web-based survey will be developed for all proposed courses and be distributed at the end of the semester to collect student feedback. * Student perception and attitude toward no cost materials including: ratings of the no cost materials used in this course comments and suggestions for course improvements 3. Student evaluation of the instructor. Formal student evaluation of the instructor can also provide information about teaching effectiveness using no cost materials. This evaluation is based on standardized forms for every course. For each of the measurement, the investigators are going to conduct two levels of analysis: 1) comparing the achievement levels of the course learning outcomes - generally, 75% is the aimed passing rate in undergraduate courses, and, 2) comparing the achievement levels to those from past offerings where costly textbooks were used. The investigators will use the data from the sections taught in the past 2 years. In

addition, Kennesaw State University requires all online courses to be reviewed and approved following an internal review process using Quality Matters (QM) standards. This review will insure the no-cost learning materials used or developed for the 5 required IT courses are instructionally sound. The College of Computing and Software Engineering will also conduct subject matter expert reviews for all developed courses to ensure the quality of the learning materials.

Timeline:

Spring 2018

Collect baseline statistics on each course (course developers – those faculty who are in charge of the course for this study)

Course modules redesigned to use the no cost materials. These include all new content, readings, lecture notes, video clips, exercises, labs, and assignments. The changes are reflected in the learning module study guides. (completed by course developers)

Course level assessment and informational materials redesign. This includes quizzes, tests, and syllabus. (course developers and instructional designer)

Submit the developed courses for instructional design review through Quality Matters. (instructional designer and KSU Distance Learning Center office)

Submit the developed courses for subject matter expert review. (department Chair)

Summer 2018

Develop a survey on effectiveness of the no cost materials (all course developers and instructional designer)

Teach:

IT 3123 – hardware/Software, Dr. Rutherford

CSE 2300 – Discrete Structures, Prof. Rutherford

Survey two summer courses and give student course evaluation (course developers and instructional designer)

Fall 2018

Teach:

IT 3223 – Software Acquisition and Proj. Management, Prof. VandeVen

IT 4683 – Management Information Technology & HCI, Dr. Halstead-Nussloch

IT 4723 – IT Policy and Law, Prof. Tatum

Survey three fall courses and give student course evaluation (course developers and instructional designer)

Complete final assessment data analysis and prepare a final report (all course developers and instructional designer)

Budget:

The funding mainly compensates our team of investigator's work and activity beyond normal teaching load or other job responsibilities in order to successfully complete the project. For each proposed course, course developers approximately will spend at least 80 hours in developing the no-cost learning material and be the instructor of record, and, will spend 20 hours in course assessment. Instructional support will devote at a minimum 50 hours in assisting course developers. Thus, we request the budget of this project as follows.

Dr. Rebecca Rutherford, Project lead; course developer and instructor of record of IT 3123, \$5,000

Prof. Susan VandeVen, course developer and instructor of record for IT3223, \$5,000

Dr. Richard Halstead-Nussloch, course developer and instructor of record for IT 4683, \$5,000

Prof. Dawn Tatum, course developer and instructor of record for IT 4723, \$5,000

Prof. James Rutherford, subject matter expert, course developer and instructor of record for CSE 2300, \$5000

Dr. Zhigang Li, Provide Instructional Design Support to all five proposed courses, \$1500

Travel: \$3500, for project team members to attend the ALG kickoff and subsequent meetings to bring back information to the team members. Our project team is also planning to submit a paper to reputable IT education conference such as ACM SIGITE 2018 (Special Interest Group in IT Education). Travel money will be used to attend conferences to present findings from the grant.

Total Budget: \$30,000

Only open source software or free software will be used in this project thus there is no additional spending on software or equipment purchasing.

Sustainability Plan:

The IT department implemented a course coordinator/developer system for all courses. A course coordinator/developer updates course content based on research, publications and feedback from faculty, students, alumni and our Industrial Advisory Board. Each of the investigators, except the instructional designer, is a course coordinator/developer for their corresponding course. A course coordinator/developer creates and maintains the course

materials and teaching plans. He/she also teaches the course at least once a year to make sure all resources are valid and makes necessary changes and updates. This makes sure all no-cost materials and resources are highly sustainable in the future offerings of this course. The coordinator/developer also brings major/minor course changes to the annual assessment retreat for all IT faculty.

Final Semester of Spring 2017
Instruction:



January 19, 2018

ALG Grant Committee
University System of GA

Dear Colleagues:

This letter is in support of the Proposal “Staying Current in Information Technology- Transforming Required Undergraduate IT Courses” submitted from Kennesaw State University, Information Technology department faculty. As Department Chair for Information Technology, I clearly see the need for bringing down costs for our students. The ALG grants assist faculty to prepare no-cost courses that allow students to take courses without the monetary burden of expensive textbooks.

Several faculty in the Information Technology Department at Kennesaw State University have successfully carried out ALG grants for several of our undergraduate Information Technology courses. The current proposal addresses five of our required undergraduate courses in the IT curriculum. The savings already realized from the previous ALG grants encouraged our faculty to develop this new ALG grant proposal to help our students save even more money.

I strongly support this proposal. This is a very sustainable proposal as we have two Information Technology undergraduate degree programs. Many of our students take courses online as well as in-class. Creating the no-cost for textbook version of our five required undergraduate IT courses will allow students for many years to realize savings from not buying textbooks. As Information Technology material is constantly changing, the concept of not relying on just textbooks for courses is extremely important to our field.

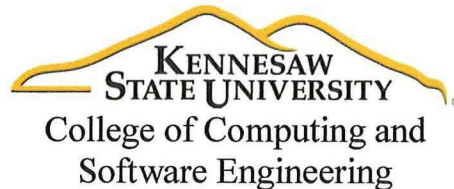
This is a very solid proposal. All faculty participating in the previous ALG grants completed their courses and offered them successfully. Papers for several conferences, and workshops about the previous grants have been created and presented. This concept has been well received in the information technology academic community. I believe that this new ALG proposal will have the same student satisfaction and success that the previous ALG grants did. This new proposal will have a unique impact as it addresses HIT courses. Thank you for your consideration for this proposal.

Sincerely,

Rebecca H. Rutherford, Ed.D.

Interim Assistant Dean of the College of Computing & Software Engineering, Department
Chair for Information Technology, Professor of Information Technology

brutherf@kennesaw.edu



January 19, 2018

Dear Affordable Learning Georgia (ALG) Grant Reviewers,

It is my pleasure to write this letter in support of the proposal titled “Staying Current in Information Technology-Transforming Required IT Courses” submitted by Drs. Rutherford, Halstead-Nussloch, Li, and Ms. Tatum, Ms. VandeVen, and Mr. Rutherford from our Information Technology (IT) Department at Kennesaw State University.

In this project, the primary investigators will work as a team to replace existing, costly textbooks in five undergraduate information technology courses with no-cost-to-students learning materials. Their efforts will significantly lower the cost of education for students, saving over \$144k per year and impacting over 1000 students per year at KSU. Additionally, this will generate a positive impact on the retention, progression, and graduation for the College of Computing and Software Engineering. Additionally, given the rapid change of the IT field, having digital materials available to students will improve the ability to keep them updated with the latest advances in the field of information technology.

The proposers have past experience with a successful ALG projects, thus the quality and success of this project is highly likely. The investigators in this project are also designated course architects who are responsible for the development and the maintenance of the to-be-transformed courses.

In conclusion, I wholeheartedly support this effort to improve access to our IT program. This proposal has the full support of the College of Computing and Software Engineering.

Sincerely,

Dr. Jon A. Preston
Interim Dean
College of Computing and Software Engineering
Kennesaw State University

**Affordable Learning Georgia Textbook Transformation Grants
Rounds Ten and Eleven
For Implementations beginning Spring Semester 2018
Running Through Fall Semester 2018**

Proposal Form and Narrative

Submitter Name	Rebecca H. Rutherford
Submitter Title	Department Chair, Professor of Information Technology
Submitter Email	brutherf@kennesaw.edu
Submitter Phone Number	470-578-7399
Submitter Campus Role	<i>Proposal Investigator (Primary)</i>
Applicant Name	<i>Rebecca Rutherford</i>
Applicant Email	brutherf@kennesaw.edu
Applicant Phone Number	470-578-7399
Primary Appointment Title	<i>Department Chair, Professor of Information Technology</i>
Institution Name(s)	Kennesaw State University
Team Members	<p>Dr. Becky Rutherford - brutherf@kennesaw.edu</p> <p>Dr. Rich Halstead-Nussloch - rhalstea@kennesaw.edu</p> <p>Prof. Dawn Tatum - dtatum7@kennesaw.edu</p> <p>Prof. Susan VandeVen - svandev@kennesaw.edu</p> <p>Prof. Jim Rutherford - jruther3@kennesaw.edu</p> <p>Dr. Zhigang Li - zli8@kennesaw.edu</p>

Sponsor, Title, Department, Institution	<p>Dr. Rebecca H. Rutherford</p> <p>Interim Assistant Dean of the College of Computing & software Engineering, and Department Chair, Information Technology</p> <p>Information Technology Department</p> <p>Kennesaw State University</p>				
Proposal Title	<p>Staying Current in Information Technology-Transforming Required Undergraduate IT Courses</p>				
Course Names, Course Numbers and Semesters Offered	<p>IT 3123 - Hardware/Software Concepts- every semester - 3 fall, 3 spring, 2 summer</p> <p>IT 3223 - Software Acquisition & Project Management- every semester - 3 fall, 3 spring, 2 summer</p> <p>IT 4683 - Management of IT & Human Computer Interaction- every semester - 2 fall, 2 spring, 2 summer</p> <p>IT 4723 - IT Policy and Law- every semester, every semester - 2 fall, 2 spring, 2 summer</p> <p>CSE2300 - Discrete Structures- every semester - 4 fall, 4 spring, 2 summer</p>				
Final Semester of Instruction	<p>Fall 2018</p>				
Average Number of Students Per Course Section	<p>29.6</p>	Number of Course Sections Affected by Implementation in Academic Year	<p>38</p>	Total Number of Students Affected by Implementation in Academic Year	<p>1125</p>
Average Number of Students Per Summer Semester	<p>216</p>				

Average Number of Students Per Fall Semester	420
Average Number of Students Per Spring Semester	425
Award Category (pick one)	<input checked="" type="checkbox"/> No-or-Low-Cost-to-Students Learning Materials <input type="checkbox"/> Specific Core Curriculum Courses
Are you planning on using an OpenStax textbook?	<input type="checkbox"/> Yes <input checked="" type="checkbox"/> No

List the original course materials for students (including title, whether optional or required, & cost for each item)	Course	Textbook Used	Cost per Student
	IT3123	IT 3123, The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology Approach, Englander, 5th edition, John Wiley and Sons, 2014; ISBN-13:978-1-118-32263-5; required;	\$150.00
	IT3223	IT 3223, a) Guide to Software Development, Springer Pub., ISBN 978-1-4471-2299-9; required; cost: \$101.20; total cost: \$32,384. b) Fundamentals of Project Management, 4th edition, AMACON; ISBN 978-0-8144-1748-5; required; cost: \$18.75	\$119.95
	IT4683	IT 4683, Using MIS 2017, Kroenke, 10th edition, ISBN 978-0-1346-0699-6; required;	\$223.15
	IT4723	IT 4723, The Legal Environment of Business and Online Commerce, 8th edition, Cheeseman, Prentice-Hall, ISBN: 978-013-397-3310; cost: \$148.15; yearly enrollment 200; total cost: \$29, 630.	\$148.15
	CSE2300	CSE 2300, Discrete Mathematical Structures, 6th ed, Pearson, ISBN: 978-0-13-469644-7; cost: \$94.97	\$94.97
	Total:		\$736.22
<i>[Material Title, optional or required]</i>			
Requested Amount of Funding	\$30,000		
Original Per Student Cost	\$736.22		

Post-Proposal Projected Per Student Cost	\$0
Projected Per Student Savings	\$736.22
Projected Total Annual Student Savings	\$144,324.50

NARRATIVE

1.1 PROJECT GOALS

In this project, we propose to take a department-wide effort to transform five required undergraduate Information Technology major courses using no-cost-to-students learning material. This project not only aims to reduce the financial burden imposed by high cost of textbooks, but also strives to develop free and open-access learning materials that offer equivalent or better educational effectiveness than traditional textbooks. These courses will then be sent through the KSU online course review process using the Quality Matters rubric to meet institutional standards of excellence as the Information Technology degree can be completed face-to-face or completely online.

Goals:

1. Transform five required undergraduate IT major courses using no-cost-to-students learning materials.
2. Create Quality Matters “ready” courses to meet institutional standards of excellence for face-to-face and online courses.

1.2 STATEMENT OF TRANSFORMATION

Research

According to Priceonomics (<http://priceonomics.com/which-major-has-the-most-expensive-textbooks/>), an average undergraduate student annually spends \$1,200 on textbooks. In addition, out of 31 majors at the University of Virginia, Computer Science (and IT) comes in 8th for the most expensive books. On the other side, the University of Virginia reports that Computer Science (and IT) textbooks only have a 25% resale value based on the original price. The highest resale value for other majors is up to 70%.

Previous ALG Grant Information

One team member was part of the round two of an "Affordable Learning Textbook Transformation Grant" in 2015 (round two, award #119). They designed and evaluated the effectiveness of no-cost-to-students learning materials for database courses in the IT department, and saved students \$110,419. The assessment results showed that the developed free material offered equivalent or better learning experience than the textbooks did. The preliminary results of the grant were published in the Proceedings of Southern Association for Information Systems Conference (SAIS 2016), the final results were published in the Proceedings of the ACM Special Interests Group in IT Education (SIGITE 2016), "Transforming IT Education with No-Cost Learning Materials". They also hosted a panel discussion on no-cost learning material in IT education, at SIGITE in October 2016. The panel attracted a lot of attention among computing faculty. Many colleagues from different states were impressed with the USG initiative and with course material developed by the team. Building on our past success and lessons learned from the prior ALG grant, we will continue our transformation efforts by developing no-cost learning material for five required undergraduate IT courses.

The Stakeholders

There are two primary sets of stakeholders for this proposal – the students taking the five required IT classes (both in-class and online students), and the faculty developing and teaching those courses. The high cost of textbooks puts a large financial burden on students and may become a road-block for students' ability to finish their education. Our team of investigators strives to make higher education more affordable to the students. The information technology required courses listed for this grant proposal have resources that are publicly accessible, free, or with an open license to use. These materials include open and free tutorials, books, videos, labs, software, and services. One of the major problems with using regular textbooks for IT courses is that information technology material is constantly changing. Textbook publishing cannot keep up with these fast changes in the technology field. In addition, tools and software packages that are part of a textbook also become obsolete. As soon as a new version of a tool or software package is released, the instructions in a textbook become obsolete. Therefore, we need to include the latest available tools to prepare hands-on labs. Digital delivery of the learning materials makes it easier to keep the content up-to-date. Developing and assembling a set of learning materials for major courses is a unique approach. It will allow us to better align the learning material not only

with the outcomes of each course, but also with the outcomes of the Information Technology program.

Compared to traditional textbooks, the open source software and web resources have many benefits: 1) the Web resources are generally free to use; 2) they are constantly being updated and always reflect the latest trends and industrial development; and, 3) the materials from the Web are also more dynamic and interactive. The pitfalls of Web resources are that they are often disorganized and may contain inaccurate information. However, members of our team of investigators are not only subject matter experts in the information security field, but also proficient educators who on average have more than 10 years teaching experience including online teaching. We will select, organize and integrate resources from the web and transform the information into instructionally sound learning materials for the proposed courses including content that the team members develop themselves. We strongly believe that the new learning materials will offer up-to-date, equivalent or better learning effectiveness compared to the original textbooks. Digital delivery also allows us to add interactive elements into the learning materials. The interactive content will not only engage the students, but also improve their learning experience. It will help to enhance the learning outcomes and learning satisfaction.

The Impact

The impact of our transformation efforts will be profound. By our estimates, more than 1125 students will benefit from the no-cost learning material each year. Moreover, it will benefit more students in the Bachelor of Science in Cybersecurity (eMajor) approved by the Board of Regents. One of the required courses proposed for this grant is also part of the BS in Cybersecurity. Student numbers are not included for the cybersecurity degree in this grant, but the expectation is that there will be an additional 120 students for this course per year within two years. The goal of eMajor is to reduce the cost of education by using prior learning assessments, lower tuition and potentially no-cost learning materials (<https://emajor.usg.edu>). The proposed project is expected to save current students \$144,324.50 in textbook costs each year (not counting the cybersecurity savings).

Because of the cost savings from not having to buy textbooks, students may be able to take a few more courses each year and graduate sooner. Having a series of required IT courses adopting no-cost-to-student material not only offers better and more consistent learning experience to students, but also makes our nationally renowned IT programs more affordable. As a result, our IT programs could recruit more students and produce more qualified IT professionals that Georgia needs. Our experience gained in this transformation project could be useful to other programs or departments who want to lower the cost of education to their students in IT programs across Georgia. In summary, we believe the proposed project will have a positive impact in students' retention, progression, and graduation at program, department and institution levels.

As shown in the following table, the textbooks used in the five required IT undergraduate major courses are expensive. In fact, most textbooks used in Information Technology are costly in general. In addition, due to the fast evolving nature of the technology field, the textbooks used in the proposed courses are updated frequently, which negatively impacts their resale value to the

students. The goal of our transformation is to replace the textbook used in the proposed courses with no-cost-to-students learning materials that offer equal or higher educational effectiveness.

Data

Table 1: Enrollments and Projected 2018 Enrollments of 5 IT courses

Course	Spring 2017	Summer 2017	Fall 2017	Total	Projected 2018 Enrollment	
					Number of Sections	Total Number of students
IT3123	93	40	78	211	8	225
IT3223	112	39	84	235	8	245
IT4683	0	41	41	82	6	90
IT4723	50	38	47	135	6	140
CSE2300	170	58	170	398	10	425
Total	425	216	420	1061	38	1125

As shown in the following table, the textbooks used in the five required IT undergraduate major courses are expensive. In fact, most textbooks used in Information Technology are costly in general. In addition, due to the fast evolving nature of the technology field, the textbooks used in the proposed courses are updated frequently, which negatively impacts their resale value to the students. The goal of our transformation is to replace the textbook used in the proposed courses with no-cost-to-students learning materials that offer equal or higher educational effectiveness.

Table 2: Costs of Current Textbooks for 5 IT Courses

Course	Textbook Used	Cost per Student	Projected Enrollment	Projected Costs
IT3123	IT 3123, The Architecture of Computer Hardware, Systems Software, and Networking: An Information Technology Approach, Englander, 5th edition, John Wiley and Sons, 2014; ISBN-13:978-1-118-32263-5; required;	\$150.00	225	\$33,750
IT3223	IT 3223, a) Guide to Software Development, Springer Pub., ISBN 978-1-4471-2299-9; required; cost: \$101.20; total cost: \$32,384. b) Fundamentals of Project Management, 4th edition, AMACON; ISBN 978-0-8144-1748-5; required; cost: \$18.75	\$119.95	245	\$29,387.75
IT4683	IT 4683, Using MIS 2017, Kroenke, 10th edition, ISBN 978-0-1346-0699-6; required;	\$223.15	90	\$20,083.50
IT4723	IT 4723, The Legal Environment of Business and Online Commerce, 8th edition, Cheeseman, Prentice-Hall, ISBN: 978-013-397-3310; cost: \$148.15; yearly enrollment 200; total cost: \$29, 630.	\$148.15	140	\$20,741
CSE2300	CSE 2300, Discrete Mathematical Structures, 6th ed, Pearson, ISBN: 978-0-13-469644-7; cost: \$94.97	\$94.97	425	\$40,362.25
	Total:	\$736.22	1125	\$144,324.50

1.3 TRANSFORMATION ACTION PLAN

With a coordinated effort, our team of investigators plan the following activities to transform 5 required Information Technology courses to completely use no-cost learning materials:

- Research and identify no cost reading materials for each of the learning modules in each course. The reading list includes both required readings and optional readings. All of these readings will be publicly accessible, free to use, or openly licensed.
- Research and identify no cost materials that can be shared across the courses.
- Develop study guides and lecture notes for students' use to review course content and key learning points.
- Adopt or develop content, assignments, exercises and lab materials that are no cost to students to replace the ones in the textbooks.
- Develop test banks to replace the ones in the textbooks.
- Adopt open source or no-cost-to-student lab ware for students to gain hands-on experience.
- Update the syllabus to include major resources and no cost materials.
- Re-develop the proposed courses in our learning management system, D2L Brightspace, following Quality Matters™ standards and get the course approved for online instruction.

The responsibilities of each investigator is described as follows.

Dr. Rebecca Rutherford, IT 3123, Project lead; Subject matter expert, course developer and instructor of record of IT 3123.

Prof. Susan VandeVen, IT 3223, subject matter expert, course developer and instructor of record for IT 3223.

Dr. Richard Halstead-Nussloch, IT 4683, subject matter expert, course developer and instructor of record for IT 4683.

Prof. Dawn Tatum, IT 4723, subject matter expert, course developer and instructor of record for IT 4723.

Prof. James Rutherford, CSE 2300, subject matter expert, course developer and instructor of record for CSE 2300.

Dr. Zhigang Li, Provide Instructional Design Support to all five proposed courses.

All course design with the no-cost materials will be provided through D2L Brightspace for our students and on the ALG website for the public access.

1.4 QUANTITATIVE AND QUALITATIVE MEASURES

The investigators plan to assess the effectiveness of our proposal in two ways. Qualitatively, we will design a survey and gather inputs from the students after they use the no-cost learning material. Quantitatively, we will compare students' performance data gathered from sections using traditional textbooks and sections using no-cost learning material.

The investigators will collect student performance data such as pass rates from the five proposed courses taught with a textbook by team members for spring, summer and fall 2017. This data will be used as a baseline for comparison of student performance in courses with alternative no cost material. Our assessment plan can be summarized as follows.

1. Student performance measures. This data is from the overall class performance based on the grading of student works. Metrics include:

- * Class average, grades distribution, pass rate for each grading item.
- * Overall letter grades distribution, pass rate, withdraw rate, and fail rate.
- * Percentage of students meeting or exceeding learning outcomes

2. Specific survey on no-cost learning materials. A web-based survey will be developed for all proposed courses and be distributed at the end of the semester to collect student feedback.

- * Student perception and attitude toward no cost materials including:
 - ratings of the no cost materials used in this course
 - comments and suggestions for course improvements

3. Student evaluation of the instructor. Formal student evaluation of the instructor can also provide information about teaching effectiveness using no cost materials. This evaluation is based on standardized forms for every course.

For each of the measurement, the investigators are going to conduct two levels of analysis: 1) comparing the achievement levels of the course learning outcomes - generally, 75% is the aimed passing rate in undergraduate courses, and, 2) comparing the achievement levels to those from past offerings where costly textbooks were used. The investigators will use the data from the sections taught in the past 2 years.

In addition, Kennesaw State University requires all online courses to be reviewed and approved following an internal review process using Quality Matters (QM) standards. This review will insure the no-cost learning materials used or developed for the 5 required IT courses are instructionally sound. The College of Computing and Software Engineering will also conduct subject matter expert reviews for all developed courses to ensure the quality of the learning materials.

1.5 TIMELINE

Spring 2018

- Collect baseline statistics on each course (course developers – those faculty who are in charge of the course for this study)
- Course modules redesigned to use the no cost materials. These include all new content, readings, lecture notes, video clips, exercises, labs, and assignments. The changes are reflected in the learning module study guides. (completed by course developers)
- Course level assessment and informational materials redesign. This includes quizzes, tests, and syllabus. (course developers and instructional designer)
- Submit the developed courses for instructional design review through Quality Matters. (instructional designer and KSU Distance Learning Center office)
- Submit the developed courses for subject matter expert review. (department Chair)

Summer 2018

- Develop a survey on effectiveness of the no cost materials (all course developers and instructional designer)
- Teach:
 - IT 3123 – hardware/Software, Dr. Rutherford
 - CSE 2300 – Discrete Structures, Prof. Rutherford
- Survey two summer courses and give student course evaluation (course developers and instructional designer)

Fall 2018

- Teach:
 - IT 3223 – Software Acquisition and Proj. Management, Prof. VandeVen
 - IT 4683 – Management Information Technology & HCI, Dr. Halstead-Nussloch
 - IT 4723 – IT Policy and Law, Prof. Tatum
- Survey three fall courses and give student course evaluation (course developers and instructional designer)

- Complete final assessment data analysis and prepare a final report (all course developers and instructional designer)

1.6 BUDGET

The funding mainly compensates our team of investigator's work and activity beyond normal teaching load or other job responsibilities in order to successfully complete the project. For each proposed course, course developers approximately will spend at least 80 hours in developing the no-cost learning material and be the instructor of record, and, will spend 20 hours in course assessment. Instructional support will devote at a minimum 50 hours in assisting course developers. Thus, we request the budget of this project as follows.

Dr. Rebecca Rutherford, Project lead; course developer and instructor of record of IT 3123, \$5,000

Prof. Susan VandeVen, course developer and instructor of record for IT3223, \$5,000

Dr. Richard Halstead-Nussloch, course developer and instructor of record for IT 4683, \$5,000

Prof. Dawn Tatum, course developer and instructor of record for IT 4723, \$5,000

Prof. James Rutherford, subject matter expert, course developer and instructor of record for CSE 2300, \$5000

Dr. Zhigang Li, Provide Instructional Design Support to all five proposed courses, \$1,500

Travel: \$3,500, for project team members to attend the ALG kickoff and subsequent meetings to bring back information to the team members. Our project team is also planning to submit a paper to reputable IT education conference such as ACM SIGITE 2018 (Special Interest Group in IT Education). Travel money will be used to attend conferences to present findings from the grant.

Total Budget: \$30,000

Only open source software or free software will be used in this project thus there is no additional spending on software or equipment purchasing.

1.7 SUSTAINABILITY PLAN

The IT department implemented a course coordinator/developer system for all courses. A course coordinator/developer updates course content based on research, publications and feedback from faculty, students, alumni and our Industrial Advisory Board. Each of the investigators, except the instructional designer, is a course coordinator/developer for their corresponding course. A course coordinator/developer creates and maintains the course materials and teaching plans. He/she also teaches the course at least once a year to make sure all resources are valid and makes necessary changes and updates. This makes sure all no-cost materials and resources are highly sustainable in the future offerings of this course. The coordinator/developer also brings major/minor course changes to the annual assessment retreat for all IT faculty.

1.8 REFERENCES & ATTACHMENTS

A letter of support must be provided from the sponsoring area (unit, office, department, school, library, campus office of the Vice President for Academic Affairs, etc.) that will be responsible for receipt and distribution of funding. Letters must reference sustainability. In the case of multi-institutional affiliations, all participants' institutions/departments must provide a letter of support.

Syllabus

[CSE 2300 Course Syllabus](#)

Module 1 – Importance of Discrete Structures

The purpose of this sub-module is to introduce you to the importance of discrete structures for computing. In particular, on completion of this sub-module, you will be able to

1. Explain the term "discrete" as used in this context
2. Explain why discrete structures are so important in computing.

Please read <http://cnx.org/content/m14586/latest/?collection=col10768/latest> by They Duy Bui.

You can also watch the file "11-01-00: What kinds of problems are solved in discrete math?" from <http://www.oercommons.org/courses/discrete-mathematics/view>. This is an introductory lecture by Shai Simonson, but you will need to make sure you have RealPlayer or some other tool to display videos. Clearly, it is an introduction to the course as taught by Dr. Simonson but it does provide some idea about what discrete mathematics is and why it is important for computing.

Finally, there is some more information on pages ix and x in Thomas VanDrunen's book "Discrete Structures and Functional Programming", which you can find at <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.9659&rep=rep1&type=pdf>.

Module 2 – Propositional Logic

Logic is the study of valid reasoning and, as such, is relevant to much more than just computing. It turns out that there are many different types of logic, of which we will consider two, namely propositional and first-order predicate calculus, in some detail in this module. However, there is another logic that is of some interest in computing, namely temporal logic, and we will very briefly cover temporal logic as well.

Moreover, since this course is about discrete structures in computing, we will also discuss a few examples of how logic is relevant in computing.

As said, propositional logic is the simplest logic. It is used to reason about propositions or sentences that are either true or false but not both. Not all sentences are propositions. For example, questions are sentences but are not true or false and therefore are not propositions.

In propositional logic, we form complex propositions from simpler propositions using the connectives \sim (NOT), $\&$ (AND), \vee (OR) and \rightarrow (IMPLIES, or IF .. THEN), and we determine the truth value of a complex proposition (i.e., whether it is true or false) from the truth value of the propositions that it is composed out of. Note that different authors will use different symbols for the connectives.

Video: <http://youtu.be/-svsnPl7qcQ>

Video class lecture: <http://www.youtube.com/watch?v=-v3u1VGXc6M>

Video Class Lecture Axioms: <http://www.youtube.com/watch?v=9J-jFz9iJLM>

Free textbook chapter link: <https://cnx.org/contents/IdMjj0pQ@1.1:LhBnDMwS@1/Discrete-Structures-Logic>

Module 3: Predicate Logic

Class video: <http://www.youtube.com/watch?v=FBWS3RNsi7A>

Video: <http://youtu.be/YbNmPievBak>

Link free textbook: <https://cnx.org/contents/IdMjj0pQ@1.1:LhBnDMwS@1/Discrete-Structures-Logic>

Module 4: Logic in Computing

Declarative logic: Normally, when you write a computer program, you not only instruct the machine what you want it to achieve, you also tell it how to achieve it. Thus, the following code fragment shows how to calculate the average of three numbers:

```
double average (int num1, int num2, int num3) {
    int sum = num1 + num2 + num3;
    double avg = sum/3;
    return avg;
}
```

Note that in the above, we have to specify not only what is to be computed; we also have to specify how it is to be computed. Clearly, this is a very simple example but, as you know, for large programs this gets more and more complicated, and we typically have to comment the code to remind ourselves and others what the code is meant to achieve.

An alternative style of programming is called "declarative programming", in which we tell the program what we want it to determine for us, but leave it up to the program itself, or rather the interpreter or compiler for the programming language in question, to determine how the result is computed.

We briefly discuss two examples of declarative programming languages, namely SQL and Prolog. Depending on the major you are in, you will already have come across at least one of them or will do so in the near future.

PROLOG: Another way in which logic is used in computing is through a programming paradigm called logic programming. As you probably know, there are different approaches to creating programming languages, often referred to as programming paradigms. In this module, we will very briefly discuss programming paradigms. Programming paradigms are covered in great detail in CS3123 Programming Language Concepts.

There are four main programming paradigms, namely

- Procedural programming
- Object-oriented programming
- Functional programming
- Logic programming

In the procedural programming paradigm, a program is explicitly regarded as a sets of procedures. C is an example. In object-oriented programming, a program is seen as a collection of objects interacting with each other. Java and C++ are examples of object-oriented programming languages, although -in my opinion- poor examples. In the functional programming paradigm, a program is seen as a collection of functions in the sense in which this term is defined in mathematics and covered later in this course. Lisp is an example.

For our purposes, the programming paradigm that is more relevant is logic programming. Under this programming paradigm, programs are seen as sets of logical sentences, expressing facts and rules about some problem domain. Programming then becomes a matter of interrogating the program.

Perhaps the best known example of a programming language is Prolog. A Prolog consists of a series of facts and rules, expressed as Horn clauses. A Horn class is a universally quantified conditionals, in which there is only one statement in the consequence and the antecedent is a conjunction. The following is a very simple example of a prolog program

```
sheep(sam)
sheep(susie)
```

```
male(sam)
female(susie)
ram(X) :- sheep(X), male(X)
ewe(X) :- sheep(X), female(X)
```

The first four lines simply give some facts, while the last two code up the rules that a male sheep is a ram, and a female sheep is an ewe. Prolog uses capitals as variables. The last two prolog clauses can be rendered in the language of predicate logic that we have used as

```
( $\forall x$ )[(sheep(x) & male(x))  $\rightarrow$  ram(x)]
( $\forall x$ )[(sheep(x) & female(x))  $\rightarrow$  ewe(x)]
```

In order to interrogate a Prolog program, we simply ask the question we want the program to answer. Thus, if we want to know whether Sam is a ram, we simply type

```
?- ram(sam) .
```

and the program will eventually answer "yes".

We can also ask the program to find any rams. We would do so by using a variable:

```
?- ram(X) .
```

and the program will answer

```
X = sam.
```

One of the main problems with Prolog from a logical point of view is the way in which it deals with negation.

Prolog uses "negation-as-failure". As your exercise for this sub-module, use the internet to find out what negation-as-failure means, and give the definition and the source, and the explanation why negation-as-failure is problematic from a logical point of view, in the associated drop box.

Module 5 – Program Correctness

A final application of logic in computing in general and computer science in particular is in the area of program correctness. A program is correct when it does what it is intended to do, and it is formally correct if it can be mathematically proven to be correct.

Normally, we use testing methods to determine program correctness. Suppose that you are asked to write a program that prompts a user for a number and then calculates and displays the square of that number. Then, once written, you will test the program by inputting a number of numbers and making sure that the number that is displayed is indeed the square of the number that you input. You may also see what happens when the user enters an illegal input, such as a string.

While this type of testing is acceptable for many applications, there are cases where you will want to be more certain that the program is correct. An example might be a routine that controls a nuclear power plant. For applications such as these, one would like to be able to mathematically prove that the program is correct.

Clearly, in order to be able to formally prove that a program is correct, you need a language in which to specify very precisely what you want the program to achieve, and computer scientists and software engineers have therefore developed a number of formal specification languages, including Z (pronounced "zed", not "zee") and VDM (http://en.wikipedia.org/wiki/Specification_language). Most formal specification languages are based on formal logic.

Gödel's Incompleteness Algorithm: Logicians are interested in proving results about the various logics that they have defined. Such results are often called "meta-logical" results, as they are results *about* the logic in question.

One particular interesting meta-logical result is Gödel's first incompleteness result. The theorem states that no consistent system of axioms whose theorems can be listed by an effective procedure is capable of proving all truths about the relations of the natural numbers

(http://en.wikipedia.org/wiki/G%C3%B6del%27s_incompleteness_theorems). In other words, there is no computer program that can prove everything that is true about arithmetic.

The technique that Gödel developed to prove this result can also be used to prove that the halting problem is undecidable. The halting problem (http://en.wikipedia.org/wiki/Halting_problem) is essentially the problem of deciding, given a program and an input, whether the program will eventually halt on this input or will run for ever, and Turing proved that the halting problem is not decidable

Yet another way of reformulating the problem is in terms of predicate logic:

There is no computer program that, for any set of propositions Γ and a proposition ϕ , will be able to determine whether ϕ follows from Γ or not.

In fact, predicate logic is semi-decidable: There is a computer program that will stop and say "yes" if ϕ actually follows from Γ , but there is no program that will say "no" if it does not.

To illustrate the problem (and note this is an illustration and not a proof), consider the following proposition:

A person is Jewish if their mother is Jewish.

Assuming that we do not know directly whether Sam is Jewish or not, we could try to determine Sam's Jewishness by figuring out whether Sam's mother is Jewish. Again, assuming that we have no direct evidence to determine whether Sam's mother is Jewish, we could try to prove that the mother of Sam's mother is Jewish, and so on. You see the problem.

Clearly, we are probably still interested in creating a computer program to automate reasoning in predicate logic, often referred to as an automated theorem prover for first-order predicate calculus, but we also want to make sure that the program does not run forever in the case that the proposition we want to prove does not follow.

The solution to this problem lies in the use of heuristics, rules-of-thumb that give the right answer in most cases, but that are not guaranteed to give the right answer in all cases. The use of heuristics is prevalent in Artificial Intelligence, and one could argue that one of the reasons for the emergence of Artificial Intelligence is the fact that we are interested in finding solutions for problems for which we can prove that there are no solutions that are guaranteed to work, such as automated theorem provers for first order predicate calculus.

An example of a heuristic that we can use in our example above is to stop searching once we have applied the rule six times. In other words, once we cannot directly determine that the mother of the mother of the mother of the mother of the mother of Sam is Jewish or not, we simply stop searching and assume that she was not Jewish, and we conclude that we cannot prove that Sam is Jewish. You will see the problem: We would draw the wrong conclusion if there actually is information stretching back 10 generations along Sam's maternal lineage that the mother in question was Jewish.

Free Textbook chapter: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.9659&rep=rep1&type=pdf>

Module 6 – Algorithms

Ted Video: <https://youtu.be/6hfOvs8pY1k>

What is an Algorithm: Perhaps the most fundamental concept in computing is the concept of an algorithm. An algorithm is a finite set of unambiguous and precise instructions that

- can be executed by a computer and that
- takes an input, does some computation, and produces an output, and
- terminates.

It is an abstract computer program, if you will.

Clearly, the problem with this definition is what it means to be "executable". The concept of an algorithm is often explained by drawing an analogy with recipes (for example, a recipe to make rice-and-peas (<http://www.foodnetwork.com/recipes/bobby-flay/jamaican-rice-and-peas-recipe/index.html>). However, there are many well-written recipes in which the steps are not executable by every cook. For example, not all of us know what to do when we are told to "deglaze the pan with white wine".

Fortunately, in the context of computing, we can define what we mean by an executable step, and we discuss this in detail in the sub-module entitled "Algorithms".

Nor surprisingly, there are often many different algorithms to achieve a particular task. For example, there are at least four well-understood algorithms for sorting a list. The question therefore naturally arises whether we can compare different algorithms in some abstract, mathematical way, for example to determine whether one runs faster than an other. The answer is that we can and the branch of discrete mathematics that allows us to do so is called "Complexity Analysis". In complexity analysis we express the running time of an algorithm in terms of the size of the input. Thus, we can express the running time of a sorting algorithm in terms of the numbers of items in the list to be sorted. The sub-module "Complexity Analysis" provides more detail.

As we shall see as well, in calculating the complexity of an algorithm, we ignore a number of factors. As a result, complexity analysis is not necessarily the best way to compare two algorithms, and in some cases, we may want to turn to alternative ways of comparing algorithms. One option is to conduct a timing experiment. In a timing experiment, we implement the different algorithms in a programming language and then measure the time that each takes to run. As your term project for this course, you will conduct some timing experiments for different sorting algorithms, and compare the results from your timing experiments with the results from a complexity analysis.

Many of the concepts introduced in this module will be discussed in far greater detail in subsequent courses, including the Data Structures course and the Analysis of Algorithms course. In this module, we will merely scratch the surface, and we will for example only discuss iterative algorithms, and ignore recursive algorithms.

Term Algorithm: As said before, an algorithm is essentially an abstract computer program. As we said as well, often there are many different algorithms for achieving the same task, and we want to find some way to formally compare the different algorithms. This in turn means that we need to find some way to specify algorithms.

Since our algorithms take an input and produce an output, they are very similar to functions in the mathematical sense (and discuss in some detail below). After all, functions take an input and produce an output that is specific to that input. That is, functions always produce the same output on the same input. Moreover, since we want our algorithms to be executable by a computer, one could say that an algorithm is a computable function.

Mathematicians started to try to characterize what makes a function computable well before actual computers had been built. Thus, the earliest definition of computable functions was provided by Alan Turing in 1936 when he defined what he called a "Logical Computing Machine" and what later became known as a Turing machine (http://en.wikipedia.org/wiki/Turing_machine). Other definitions included lambda calculus, register machines and μ -recursive functions (see http://en.wikipedia.org/wiki/Computable_function for an overview and links to the various definitions that have been proposed).

The good thing about all these definitions is that they are equivalent. For example, all functions that are computable if you use Turing machines are also computable if you use lambda calculus, and vice versa, and the same applies for all other definitions. The bad thing is that it is very hard if not impossible to specify algorithms that do some real work in any of these formalisms, and we therefore need a more intuitive way of specifying algorithms, and fortunately there is.

In order to specify an algorithm, we need the following constructs:

- An assignment operator to assign a value to a variables and create an assignment statement (e.g., = in Java or C++);
- A series of operators to compare values (e.g, ==, <, >) and thus create tests;
- Some logical operators to combine tests into more complex tests (e.g., && and || in Java and C++);
- A conditional to branch depending on the outcome of a test (if-then-else);
- A way to make a sequence of statements;
- A way to repeat statements.

Module 7 – Complexity Analysis

Big O Complexity: As said, in many cases, there are different algorithms to achieve the same task, and for obvious reasons, we will want to use the best algorithm to create the computer program. Now, there are many ways to define "best". For example, one algorithm may be better than another because it is easier to understand, and hence to translate into a programming language, like Java or C++. Or an algorithm may be better because it uses less memory. However, in complexity analysis, we consider an algorithm better if it runs faster.

There are of course many factors that influence the speed of a computer program, other than just the algorithm that underlies the program and the size of the input to the algorithm. For example, the type of computer and in particular the speed of the processor in the computer have a big influence on the speed of the program, and there is a somewhat corny joke among computer scientists that the best way to speed up a program is to save it to a USB drive and wait for the next generation computer to come out.

However, since we are comparing algorithms, i.e. abstract computer programs, we can ignore most of these other factors and we only consider the size of the input, and we express the running time of an algorithm as a function of the size of the input. The notation that we use to express the complexity of an algorithm is O (big-Oh). Moreover, when we give the complexity of an algorithm, we ignore all terms other than the term that most determines the growth of the value. Thus, if we determine that an algorithm runs in $n \cdot \log(n) + 73$ time units, where n is the size of the input, we state that the complexity of the algorithm is $O(n \cdot \log(n))$. The rationale is that, as n gets sufficiently large, the contribution of the other terms to the value of the function that determines the running time becomes negligible.

Link: http://www.youtube.com/playlist?list=PL2_aWCzGMAwI9HK8YPVBJElbLb13ufctn

Determining Complexity Analysis: In the sub-module on algorithms, we saw that in order to specify an algorithm, we needed

- An assignment operator to assign a value to a variables and create an assignment statement (e.g., = in Java or C++);
- A series of operators to compare values (e.g, ==, <, >) and thus create tests;
- Some logical operators to combine tests into more complex tests (e.g., && and || in Java and C++);
- A conditional to branch depending on the outcome of a test (if-then-else);
- A way to make a sequence of statements;
- A way to repeat statements, for which we used iteration (e.g., for-loops).

We can use these constructs to determine the time-complexity of an algorithm by using the following rules:

- The complexity of an assignment, a test and a combination of test is constant, i.e. it is $O(1)$.
- The complexity of an if-then-else statement is the maximum of the complexity of the then-part and the else-part. Thus,
$$O(\text{if } \langle \text{some_test} \rangle \{ \langle \text{then-part} \rangle \} \text{ else } \{ \langle \text{else-part} \rangle \}) = \text{MAX}(O(\langle \text{then-part} \rangle), O(\langle \text{else-part} \rangle))$$
- The complexity of a sequence of statement is the maximum of the complexity of the statements in the sequence. Thus,
$$O(\langle \text{statement-1} \rangle; \langle \text{statement-2} \rangle; \dots ; \langle \text{statement-n} \rangle;) = \text{MAX}(O(\langle \text{statement-1} \rangle), O(\langle \text{statement-2} \rangle), \dots, O(\langle \text{statement-n} \rangle))$$
- The complexity of a for loop is the complexity of the body of the loop times the number of times we go through the loop. Thus,
$$O(\text{for}(i = 0; i < n; i++) \{ \langle \text{body} \rangle \}) = n * O(\langle \text{body} \rangle)$$

Watch the videos "Time complexity analysis - How to calculate running time?" and "Time complexity analysis - some general rules" at http://www.youtube.com/playlist?list=PL2_aWCzGMAwI9HK8YPVBjElbLbI3ufctn for more details and some examples. As before, you will have to watch some adverts. Also, the video "Time complexity analysis - some general rules" mentions asymptotic notations and in particular theta. You can understand the material without knowing additional details about these concepts, but if you are interested watch the video "Time complexity analysis: asymptotic notations - big oh, theta ,omega" at the same site.

Video Complexity Analysis: http://www.youtube.com/playlist?list=PL2_aWCzGMAwI9HK8YPVBjElbLbI3ufctn

Module 8 – Sets

Free textbook chapter: <http://cnx.org/content/m15772/latest/?collection=col10768/latest>

Video on Sets: https://youtu.be/t3XdRbPNtdg?list=PLUpS0WwSvA3e7HtgzNHMivo0T8V0etX_Z

Paradox of Sets: The material referenced above covers what Duy Bui calls "naive set theory". There is a complication with naive set theory, which was first published by Bertrand Russell in 1901, although the problem was known by some mathematicians before. Russell constructed the set of all sets that are not a member of themselves, i.e.

$$\{ x \mid x \notin x \}$$

and then asked whether this set is a member of itself. In other words, is

$$\{ x \mid x \notin x \} \in \{ x \mid x \notin x \}?$$

A little thought will show the problem. If this set, let's call it R , is a member of itself, then it must satisfy the condition of not being a member of itself. In other words,

$$R \in R \rightarrow R \notin R$$

On the other hand, if R is not a member of itself, then it satisfies the condition of being a member of R . In other words

$$R \notin R \rightarrow R \in R.$$

We therefore have a paradox that can only be resolved by complicating naive set theory. You can read more about Russell's paradox and how mathematicians have dealt with it at http://en.wikipedia.org/wiki/Russell%27s_paradox.

Link to Paradox: http://en.wikipedia.org/wiki/Russell%27s_paradox.

Relations: One of the topics covered in the material on relations is orders. You will also recall that creating an ordered list of items is central to the term project in that you were asked to implement two different sorting algorithms and compare their performance in a timing experiment.

The question may arise why sorting is so important. The reason is that if you can sort a set of items, and you can access any position in the set in a constant time, the complexity of determining whether a given item is an element of that set goes down from $O(n)$ to $O(\log(n))$.

Clearly, if your set is not ordered (or if you cannot access any position in the set in a constant time), then the only way in which you can determine whether a given item is an element of the set is to compare it with the first element in the set. If they are identical, you are done; if they are not, you compare it with the next element in the

set, and so on. If the item is not an element of the set, then we will need to compare the item with every element in the set, and we therefore make n comparisons where n is the number of elements in the set. If the item is an element of the set, then, on average, the item you are looking for will be somewhere half way down the list, and you will therefore have to make on average $1/2n$ comparisons. In other words, you can expect to have to make $1/2n$ comparisons. Since we ignore the constant, the complexity of this type of search, which is called "linear search" is $O(n)$.

However, if the set is ordered, and we can access any position in the set in a constant time, we start by comparing the item with the element in the middle of the set. If the item is identical to the element in the middle of the set, then we are done; if it is not, and the item is smaller than the one in the middle, we know that the item, if it is an element of the set, will be in the first half of the list; if it is larger, then it will be in the second half. So, if it is smaller, we repeat the process for the lower half of the list; if it is larger then we repeat the process for the larger half of the list. Since we keep dividing the list we are searching in half, and we can divide a list of size n in half $\log(n)$ times, and the complexity of the algorithm, which is called "binary search", is $O(\log(n))$.

Link: <http://www.youtube.com/watch?v=wNVCJj642n4>

Class Lecture: <https://youtu.be/YlhJKoqzcdY>

Module 9 – Relations

Free Textbook chapter: <http://cnx.org/content/m15775/latest/?collection=col10768/latest>

Class lecture: <https://youtu.be/ikIyUks8HRI>

Video on Relations: http://www.youtube.com/watch?v=q3Z7PiW8FNg&list=PL_D1rGgPr31PjDJPnnsyDJo1eWweVeq03

Video 2 on Relations: http://www.youtube.com/watch?v=h34hZ_hynzE

Video 3 on Relations: http://www.youtube.com/watch?v=hM_iObXeno0

Database and relations: In the module on logic, we encountered an example of declarative programming in SQL. You will recall that SQL is the most widely used language to interrogate databases.

There have been different models of building databases, but the model most widely used at the moment is the relational model, which was due to Edgar F. Cobb. In the relational model, a database is a set of relations in the sense in which we defined the term in this module, also called tables. You can find more details on the relational model at

http://en.wikipedia.org/wiki/Relational_model,

and the model will be discussed in much greater detail in the database course.

Database researchers have also created two formal languages for creating, interrogating and analyzing databases, namely relational calculus and relational algebra. You can find more details about relational algebra at

http://en.wikipedia.org/wiki/Relational_algebra

and about relational calculus at

http://en.wikipedia.org/wiki/Relational_calculus

You will see that relational calculus has two different flavors, namely tuple relational calculus http://en.wikipedia.org/wiki/Tuple_relational_calculus

and domain relational calculus (http://en.wikipedia.org/wiki/Domain_relational_calculus

Module 10 – Functions

Free Textbook chapter: <http://cnx.org/content/m15776/latest/?collection=col110768/latest>

Class lecture: <https://youtu.be/kykmBB74-HQ>

Functional Programming: Under the functional programming paradigm, a program consists of a set of functions in the mathematical sense of the word. The language ML that is used in Thomas VanDrunen's book is an example of a functional programming language. Another -widely used- functional programming language is LISP. LISP has a very simple syntax. Here is an example of a simple LISP program

```
(defun mult_through_add (a, b)
  (if (eq a 0)
      b
      a + mult_through_add(a, b - 1)
  )
)
```

`defun` defines a new function and `if` is an in-built function in LISP which takes three arguments. It evaluates the first argument and if it does not evaluate to false, it evaluates the second argument and otherwise it evaluates the third. In other words, it is the if-then-else operator. Clearly, the definition assumes that `b` is a positive number.

Once we have defined a function, we can either call it directly, or use it in the definition of another function.

Find out more details about LISP at

http://en.wikipedia.org/wiki/Lisp_%28programming_language%29

or

<http://www.gigamonkeys.com/book/>

and see an example of a larger LISP program at

<http://www.csc.villanova.edu/~dmatuszc/resources/lisp/lisp-example.html>

Module 11 – Graphs

Class Lecture: <https://youtu.be/vOomN71xYIg>

Video on Graphs: <http://www.youtube.com/watch?v=HmQR8Xy9DeM>

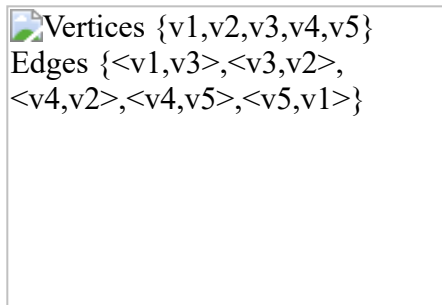
Video 2 on Graphs: <http://www.youtube.com/watch?v=cOB85BQ8gX0>

Video 3 on Graphs: <http://www.youtube.com/watch?v=0t3i30T2NB0>

Graphs as Networks: It will not come as a surprise that graphs are used heavily in order to analyze networks. In this module, we will introduce some examples and show how the various properties of graphs that were introduced in the previous sub-module and some of the algorithms that we will introduce can be used to determine properties of networks that are relevant to some applications.

Connectedness - Graphs, perhaps not surprisingly, can prove extremely useful as a tool to analyze computer networks. A good computer network is a connected graph. Given a set of network nodes and connections, which we can obviously analyze as a set of vertices and edges, a useful algorithm would be one that can quickly determine which the graph is connected.

There are different ways in which we can determine whether a graph is connected. One is through search: We can start at an arbitrary vertex, use a graph search algorithm, and count all the vertices we can reach. If the number of vertices we can reach is equal to the number of vertices in the graph, then the graph is connected. There are different graph search algorithms. Two that are particularly useful are breadth-first and depth-first search. Find out more about breadth first and depth first search, and use the dropbox "Graph Search" to show the order in which you would visit the vertices in the graph below, starting at vertex v4.



Vertices {v1,v2,v3,v4,v5}
Edges {<v1,v3>,<v3,v2>,
<v4,v2>,<v4,v5>,<v5,v1>}

Adjacency graphs are also useful tool to determine whether a graph is connected. In order to show you how this works, recall that an adjacency matrix is essentially a 2-dimensional array both of whose dimensions are the same size. We will call this the size dimension of the array.

So, to determine whether a graph is connected from its adjacency graph M, we can use the following algorithm:

```
copy M into a new graph M';
changed = 1;
while (changed == 1) {
    changed = 0;
    for (i = 0; i < size dimension of M'; i++) {
        for(j = 0; j < size dimension of M'; j++) {
            if (M'[i,j] = 1) {
                for(k = 0; k > size dimension of M', k++) {
                    if (M'[j,k] == 1 && M'[i,k] != 1)
                        M'[i,k] = 1;
                        changed = 1;
                }
            }
        }
    }
}
```

If M' now completely consists of 1, G is connected.

Link Social Networks: http://en.wikipedia.org/wiki/Social_network

Link E/R Diagrams: http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

Link Semantic Networks: http://en.wikipedia.org/wiki/Semantic_network

Link Conceptual Graphs/Networks: http://en.wikipedia.org/wiki/Conceptual_graph

Module 12 – Trees

Link: <http://www.saylor.org/site/wp-content/uploads/2011/09/CS202-Graphs1-Srini-Devadas.pdf>

Link Definition: [http://en.wikipedia.org/wiki/Tree_\(data_structure\)](http://en.wikipedia.org/wiki/Tree_(data_structure))

Video: <https://www.youtube.com/watch?v=58zSgcTj6ZQ&hd=1>

Video 2: <https://www.youtube.com/watch?v=HmQR8Xy9DeM&hd=1>

Video 3: <http://www.studyjaar.com/index.php/module/39-trees>

Binary Search Trees

- In our discussion of sorting, we saw that if a list is sorted, and you can access any item in a constant time, you can use binary search, rather than linear search, and the complexity of determining whether an item occurs in a list goes down from $O(n)$ to $O(\log(n))$. However, if you cannot access any item in a constant time, then you cannot use binary search. Fortunately, binary search trees (BSTs) solve this problem.

Link: http://en.wikipedia.org/wiki/Binary_search_tree

Video: https://www.youtube.com/watch?v=pYT9F8_LFTM&hd=1

Video 2: https://www.youtube.com/watch?v=rVU_jXyHXqw&hd=1

Video 3: <https://www.youtube.com/watch?v=pmsVttdSaVU&hd=1>

Final Report

Affordable Learning Georgia Textbook Transformation Grants

Final Report

Grant #354

To submit your Final Report, go to the Final Report submission page on the ALG website:
http://affordablelearninggeorgia.org/site/final_report_submission

Final report submission requires four files:

- This completed narrative document
- Syllabus or syllabi
 - (if multiple files, compress into one .zip folder)
- Qualitative/Quantitative Measures data files
 - (if multiple files, compress into one .zip folder)
- Photo of your team or a class of your students w/ at least one team member, minimum resolution 800x600px
 - (nearly all smartphones take photos larger than this size by default)

Follow the instructions on the webpage for uploading your documents. Based on receipt of this report, ALG will process the final payment for your grant. ALG will follow up in the future with post-project grantee surveys and may also request your participation in a publication, presentation, or other event.

General Information

Date: 12/21/2018

Grant Round: 11

Grant Number: #354

Institution Name(s): Kennesaw State University

Project Lead: Rebecca Rutherford

Team Members (Name, Title, Department, Institutions if different, and email address for each):

- Rebecca Rutherford, Interim Assistant Dean, College of Computing and Software Engineering, Department Chair for Information Technology, and Professor of Information Technology, brutherf@kennesaw.edu.
- Dawn Tatum, Senior Lecturer, College of Computing & Software Engineering, Information Technology Department, dtatum7@kennesaw.edu
- Susan VandeVen, Senior Lecturer, College of Computing & Software Engineering, Information Technology Department, svandeve@kennesaw.edu

- Richard Halstead-Nussloch, Professor of Information Technology, College of Computing & Software Engineering, Information Technology Department, rhalstea@kennesaw.edu
- James Rutherford, Senior Lecturer, College of Computing & Software Engineering, Software Engineering & Game Design Department, jruther3@kennesaw.edu
- Zhigang Li, Assistant Professor of Information Technology, College of Computing & Software Engineering, Information Technology Department, zli8@kennesaw.edu

Course Name(s) and Course Numbers:

- IT 3123 Hardware/Software: Rebecca Rutherford
- IT 4723 IT Policy & Law: Dawn Tatum
- IT 4683 Management of IT: Richard Halstead-Nussloch
- IT 3223 Software Acquisition & Project Management: Susan VandeVen
- CSE 2300 Discrete Structures: James Rutherford

Semester Project Began: Spring 2018

Final Semester of Implementation: Fall 2018

Total Number of Students Affected During Project:

Course	Enrollment
IT 3123	112
IT 4723	15
IT 4683	92
IT 3223	78
CSE 2300	238
Total	535

1. Narrative

A. Describe the key outcomes, whether positive, negative, or interesting, of your project.

Include:

- Summary of your transformation experience, including challenges and accomplishments
- Transformative impacts on your instruction
- Transformative impacts on your students and their performance

Our transformation effort is a great success. We have developed and implemented no-cost-to-student learning material for the five proposed courses. The URLs of the learning material are

listed in table one. 126 students have been impacted by our efforts. As shown in table two, students' opinions on Learning material we created are overwhelmingly positive. Our assessment data shows that, the no-cost learning material we developed are as effectively as the textbooks used in the corresponding classes.

Table 1. URL of No-Cost Learning Material

Course	URL of No-Cost Learning Material	Developer
IT 3123 Hardware/Software	http://ksuweb.kennesaw.edu/~lli13/ALG364/IT3123	Dr. Rebecca Rutherford
IT 4723 IT Policy & Law	http://ksuweb.kennesaw.edu/~lli13/ALG364/IT4723/IT4723.htm	Prof. Dawn Tatum
IT 4683 Management of IT	http://ksuweb.kennesaw.edu/~rhalstea/ALG/IT4683/index.html	Dr. Richard Halstead-Nussloch
IT 3223 Software Acq & Proj Mgt		Prof. Susan VandeVen
CSE 2300 Discrete Structures	http://ksuweb.kennesaw.edu/~lli13/ALG364/CSE2300.html	Prof. James Rutherford

Table 2. Students' Opinion on No-Cost Learning Material

Statements	Score
In general, the learning modules were organized	4.13
The content, links and other leaning module materials were sufficient to help me learn.	4.22
I liked not having to buy a textbook and instead used the materials that were provided and free.	4.45
I prefer using selected open source/free learning materials rather than a paid textbook for this course.	4.47
Overall, compared to a potential paid textbook, open resource learning materials provided the necessary assistance to learn the material.	4.62
I would take another course that uses open/free learning materials.	4.74

Note: in the survey, students are asked to express their opinion on a list of question using a 5-points scale where 1 is mostly disagree, 3 is neutral, and 5 is mostly agree.

Our plan is to get many of our undergraduate Information Technology courses completed without a textbook. The volatile area of Information Technology makes a no-textbook course ideal! Our faculty are completely onboard with the no-cost course development that the ALG grants provide.

From the instructors' perspectives, collecting and organizing the learning material ourselves not only enable us to better respond to dynamic nature of the information technology field, but also give us the flexibility to customize the course content to better serve our students. On the other hand, the transformation activities require significant efforts and time commitment from the faculty to collect, organize, create, and maintain no-cost learning material that offers equivalent learning experience as the textbooks. Our transformative efforts in replacing textbooks in the proposed courses will not happen without the strong supports from the ALG grant.

With our sustainability plan, the no-cost learning material will be continually used and hundreds and thousands of students from the Information Technology undergraduate degree Kennesaw State University will enjoy the cost savings and enhanced learning experience in the future.

B. Describe lessons learned, including any things you would do differently next time.

IT3123

What worked well: The newly designed instructor created content, along with an online free textbook assisted the students in learning the material. All of the links and videos were also important for up-to-date material for the course.

What needs to be done still: New labs will need to be added to the course when the newly created Information Technology lab goes into effect fall 2019.

IT 4683

What worked well: taking our the ISACA materials and replacing them with online links and videos for the course outcomes.

IT 4723

What worked well: creating new content for the course, updating links and videos and creating new labs for the course allowed the students to have several types of ways to learn the material.

IT 3223

What worked well: Being able to replace two books for this course saved the students quite a lot of money. Since this course looks at two major areas – software acquisition & software life cycle, and then project management, the instructor was able to find up-to-date material for both major areas of the course. Creating new course content, providing links and videos has given the students current material to meet the course outcomes.

2. Quotes

- Provide three quotes from students evaluating their experience with the no-cost learning materials.

“The IT3123 course has changed quite a bit from the previous version. I really liked having everything online (including a free textbook), and felt that all of the modules contained enough material for me to learn the outcomes of each module. I liked not having to buy a textbook.” – an IT 3123 student

“I had heard from previous students that we had to buy two books for this course, so I was surprised when we didn’t have to buy any books. This really saved me money and I still felt I could learn everything I needed to from the materials provided.” – an IT 3223 student

“The IT 4683 course seemed fine without having a textbook. I didn’t have any trouble learning the material for the course.” – an IT 4683 student

3. Quantitative and Qualitative Measures

3a. Uniform Measurements Questions

The following are uniform questions asked to all grant teams. Please answer these to the best of your knowledge.

Student Opinion of Materials

Was the overall student opinion about the materials used in the course positive, neutral, or negative?

Total number of students affected in this project: 535

1. Positive: 91.1 % of 102 number of respondents
2. Neutral: 6.45 % of 102 number of respondents
3. Negative: 2.45 % of 102 number of respondents

Student Learning Outcomes and Grades

Was the overall comparative impact on student performance in terms of learning outcomes and grades in the semester(s) of implementation over previous semesters positive, neutral, or negative?

Student outcomes should be described in detail in Section 3b.

Course	Enrollment	Student average GPA	
		Semester with no-cost material	Semester with textbooks
IT 3123	112	2.89	2.23
IT 4723	15	2.98	2.96
IT 4683	92	3.73	3.70
IT 3223	78	3.03	3.2
IT 2300	238	3.72	3.68

Choose One:

- Positive: Higher performance outcomes measured over previous semester(s)
- Neutral: Same performance outcomes over previous semester(s)
- Negative: Lower performance outcomes over previous semester(s)

Student Drop/Fail/Withdraw (DFW) Rates

Was the overall comparative impact on Drop/Fail/Withdraw (DFW) rates in the semester(s) of implementation over previous semesters positive, neutral, or negative?

Drop/Fail/Withdraw Rate:

Depending on what you and your institution can measure, this may also be known as a drop/failure rate or a withdraw/failure rate.

Course	Enrollment	Drop/Fail/Withdraw Rate Comparison	
		Current semester	Previous semester
IT 3123	112	5%	15%
IT 4723	15	8%	8%
IT 4683	92	5%	0%
IT 3223	78	12%	11%
CSE 2300	238	5%	8%

35 % of students, out of a total 535 students affected, dropped/failed/withdrew from the course in the summer and fall semesters of implementation.

Choose One:

- Positive: This is a lower percentage of students with D/F/W than previous semester(s)
- Neutral: This is the same percentage of students with D/F/W than previous semester(s)

- ___ Negative: This is a higher percentage of students with D/F/W than previous semester(s)

3b. Measures Narrative

In this section, summarize the supporting impact data that you are submitting, including all quantitative and qualitative measures of impact on student success and experience. Include all measures as described in your proposal, along with any measures developed after the proposal submission.

For this ALG proposal, we proposed to use multiple data collection methods to measure the success of our creating our no-cost courses. We looked at both quantitative and qualitative measures.

Quantitatively, we compared students' DFW rates, grades, and success in course learning outcomes. The DFW rates are taken from student registration system. The student grades and success in course learning outcomes are assessed Faculty Course Assessment Report (FCAR). Faculty in IT department at Kennesaw State University are required to create a FCAR for every course they teach for each semester. The FCAR includes students' grade and success in achieving the course learning outcomes.

Qualitatively, we developed a survey to collect students' opinion on the learning material used in the courses. Students rated their experience using a 5 points scale. Students also give the opportunities to enter comments they may have. Based on the assessment data we collected, the learning material we created offer the same level of the learning effectiveness as the textbook. Some no-cost percentages were higher than textbook courses, and some were lower. Students' performance outcomes and DFW in generally stay the same pre-implementation and post-implementation.

4. Sustainability Plan

- *Describe how your project team or department will offer the materials in the course(s) in the future, including the maintenance and updating of course materials.*

The IT department at KSU has an individual course architect architect for all courses. A course architect develops, updates and maintains course content based on research, publications and feedback from students and alumni. He/she also teaches the course at least once a year to make sure all resources are valid and make necessary changes. This makes sure all no-cost materials and resources are highly sustainable in the future offerings of this course.

5. Future Plans

- *Describe any impacts or influences this project has had on your thinking about or selection of learning materials in this and other courses that you will teach in the future.*
- *Describe any planned or actual papers, presentations, publications, or other professional activities that you expect to produce that reflect your work on this project.*

Information technology is dynamic field where existing technology frequently get updated and new technology constantly comes out. Due to this reason, the no-cost learning material model naturally fits better for IT curriculum than the traditional textbook models. The faculty in the IT department already completed several individual and transform-at-scale grants. The positive feedback from the students and our own development and implementation process inspire more faculty in the IT to get involved with developing no cost learning material for their courses.

A panel was presented at SIGITE 2018 on developing No-cost Materials for STEM fields by all of the ALG participants. Dr. Rebecca Rutherford and Prof. James Rutherford also presented a paper at the EDSIG 2018 conference on Creating No-Cost Materials for STEM Courses.

6. Description of Photograph

- *On the Final Report Submission page, you will be submitting a photo. In this document, list the names of the people shown in this separately uploaded photograph, along with their roles.*



From Left: Dr. Richard Halstead-Nussloch, Dr. Rebecca Rutherford, Prof. Datn Tatum, Professor Susan VandeVen