COURSE DESCRIPTION

Explores the traditional approach to software construction, software crisis, and software characteristics. Covers various software engineering paradigms, and the fundamental concepts of analysis, design, coding, testing and maintenance. Introduces various CASE tools.

REFERENCES AND READING MATERIALS

- Class notes and reading materials will be posted on the blackboard.
- J. Chang and D. Richardson. “Structural Specification-based Testing with ADL”. *Proceedings of


STUDENT LEARNING OUTCOMES

1. Understand the traditional approach of software development process, software characteristics, software quality, ethics issues, crisis issues and software development cost and management.
2. Understand and be able to use basic software engineering methodologies to solve large scale software/software-intensive system development. Understand and be knowledgeable about existing tools of some software engineering methodologies.
3. Be knowledgeable of software testing strategies and be able to use basic software testing technologies to validate program.

CLASS FORMAT

1. Students are expected to attend all lectures and labs.
2. Students need to silence cell phones, stop discussion, and stop doing other unrelated things during class.
3. All homework assignments must be submitted on blackboard before the end of the due date.
4. Any assignments submitted one week later than due date will have 20% cut off the grade. No assignment is accepted one week after the due date.
5. Programming assignment must be submitted with electronic copy (source code). Hard copy is not accepted and will not be graded. For the late submission, the softcopy must be emailed to me.
6. No late submission of quizzes, tests and exams.
7. All assignments must be finished on the individual basis. All identical works will result zero grade for all individuals involved.
8. Course assessment includes Pop-up tests/quizzes, tests and exams, none of them will be made up. The only excuse for missing an exam is verifiable cases of illness and emergencies and religious holy days. Please check the dates of the exams and inform me of any conflicts with religious holy days as soon as possible.
9. During tests and exams, write the answers legibly. No cell phones, pagers or calculators are allowed.
10. Cheating in any form will result in a failing grade in addition to university stipulated penalties.

PROJECT POLICIES

1. Optional project will be provided in a separated file and loaded to Blackboard.

*Dates are estimates. Examinations on dates should follow the university handbook.
2. You may be able to choose other topics you want.
3. Your project must be in reasonable size and be able to finish in 4-weeks.
4. Two deliverables of your project is required and schedule of project deliverables are available in Blackboard account.

SERVICES FOR PERSONS WITH DISABILITIES

The University provides environmental and programmatic access for persons with documented disabilities as defined in Section 504 of the Rehabilitation Act of 1973 and the Americans with Disability Act of 1990. Any student who desires information or assistance in arranging needed services for a disabling condition should contact the Director of Special Students Services, Student Center, Room 203, (256) 372-4263.

ATTENDANCE POLICY

A student is permitted one (1) unexcused absence for each credit hour generated by the class. Students need to provide university excuse for the class attendance credits back if the session is missed.

TUTORIAL ASSISTANCE

Tutorial assistance for undergraduate courses can be obtained from the Tutorial Assistance Network (TAN), a subsidiary of the Office of Academic Support Services. TAN is located in Room 100C Buchanan Hall. The telephone number is 256-372-5487.

GRADE DETERMINATION (Temporarily)

<table>
<thead>
<tr>
<th>Course Requirements</th>
<th>Points Awarded</th>
<th>Percent of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-4 assignments and one essay will be given and counted for the final grade.</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>2-4 pop-up quizzes will be given during the semester.</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>1 term project, including 1 prototype, up to 2 project presentations and demo, 1 project proposal and 2 deliverables.</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Reading Materials &amp; Presentations: 4 to 12 classic papers for the research and fundamental topics will be given. There will be 2 to 3 presentations.</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>Class participation</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>Midterm</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td>100%</td>
</tr>
</tbody>
</table>

Grading Scale

90-100  A
80-89    B
70-79    C

*Dates are estimates. Examinations on dates should follow the university handbook.
**COURSE OUTLINE (Temporary)**

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Topics</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Jan 7 – 11</td>
<td>Introduction to Software Engineering</td>
<td></td>
</tr>
<tr>
<td>Week 2</td>
<td>Jan 14 – 18</td>
<td>Software development process (chapter 1)</td>
<td><strong>Jan 21, Martin Luther King Jr. Day Holiday</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Professional ethics issues in the software</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>development process</td>
<td></td>
</tr>
<tr>
<td>Week 3</td>
<td>Jan 21 – 25</td>
<td>Agile development (chapter 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirement modeling (chapter 7)</td>
<td></td>
</tr>
<tr>
<td>Week 4</td>
<td>Jan 28 – Feb 1</td>
<td>Design Concepts (chapter 8)</td>
<td>Homework #1, references</td>
</tr>
<tr>
<td>Week 5</td>
<td>Feb 4 – 8</td>
<td>Advanced Design Concepts (chapter 9)</td>
<td></td>
</tr>
<tr>
<td>Week 6</td>
<td>Feb 11 – 15</td>
<td><strong>Quality</strong> concepts (chapter 14)</td>
<td>Presentation</td>
</tr>
<tr>
<td>Week 7</td>
<td>Feb 18 – 22</td>
<td>Software quality assurance (chapter 16)</td>
<td>Homework #2, reading assignment</td>
</tr>
<tr>
<td>Week 8</td>
<td>Feb 25 – Mar 1</td>
<td>Formal methods and verification (chapter 21)</td>
<td>Presentation</td>
</tr>
<tr>
<td>Week 9</td>
<td>Mar 4 – 8</td>
<td>Software Testing strategies (chapter 17)</td>
<td>Midterm Exam (<strong>Wednesday, March 4, 2014</strong>)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Review exam</td>
<td></td>
</tr>
<tr>
<td>Week 10</td>
<td>Mar 11 – 15</td>
<td>Spring Recess</td>
<td></td>
</tr>
<tr>
<td>Week 11</td>
<td>Mar 18 – 22</td>
<td>Software testing strategies (chapter 17)</td>
<td>Reading assignment</td>
</tr>
<tr>
<td>Week 12</td>
<td>Mar 25 – 29</td>
<td>Software testing techniques (chapter 18)</td>
<td>Homework #4</td>
</tr>
<tr>
<td>Week 13</td>
<td>Apr 1 – 5</td>
<td>White box testing (chapter 18)</td>
<td>Presentation</td>
</tr>
<tr>
<td>Week 14</td>
<td>Apr 8 – 12</td>
<td>Structure testing (chapter 18)</td>
<td>Reading assignment, Presentation (STEM Day)</td>
</tr>
<tr>
<td>Week 15</td>
<td>Apr 16 – 20</td>
<td>Project management concepts (chapter 24)</td>
<td>Review</td>
</tr>
<tr>
<td>Week 16</td>
<td>Apr 22 – 26</td>
<td>Final Exam week</td>
<td>Final Exam (TBA)</td>
</tr>
</tbody>
</table>

*Dates are estimates. Examinations on dates should follow the university handbook.*
Relationship between Course Outcomes and Program Outcomes

<table>
<thead>
<tr>
<th>Program Outcomes</th>
<th>Program Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Students will demonstrate critical knowledge, techniques, and tools of the discipline</td>
</tr>
<tr>
<td>2</td>
<td>Students will apply appropriate and emerging mathematics, computer science, and engineering technologies to solve problems</td>
</tr>
<tr>
<td>3</td>
<td>Students will work as team members and with team leaders</td>
</tr>
<tr>
<td>4</td>
<td>Students will have documented abilities for writing and presentation skills</td>
</tr>
<tr>
<td>5</td>
<td>Students will apply one or more modern computer languages to problem solving</td>
</tr>
<tr>
<td>6</td>
<td>Students will clearly express the basis for responsible and ethical behavior in their profession and recognize the need for it</td>
</tr>
<tr>
<td>7</td>
<td>Students will be able to implement concepts in software engineering, operating systems, computer architecture, and algorithm analysis</td>
</tr>
</tbody>
</table>

Program Outcomes – Mapped to Course Outcomes

<table>
<thead>
<tr>
<th>Course Outcomes</th>
<th>Program Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand the traditional approach of software development process, software characteristics, software quality, ethic issues, crisis issues and software development cost and management.</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Understand and be able to use basic software engineering methodologies to solve large scale software/software-intensive system development. Understand and be knowledgable about existing tools of some software engineering methodologies.</td>
<td>X X X X</td>
</tr>
<tr>
<td>Be knowledgable of software testing strategies and be able to use basic software testing technologies to validate program.</td>
<td>X X X X</td>
</tr>
</tbody>
</table>

Assessment Plan for the Course

Assessments consist of labs, quizzes, exams, homework assignments, and Projects. All the tests are Closed-book Closed-notes, and in-class tests.

How Data in the Course is used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

*Dates are estimates. Examinations on dates should follow the university handbook.*
Student performance data as processed by Performance Assessment Tool and student survey data are used to look into the improvement of possible topic areas and their delivery.

For a computer science program
Estimate Curriculum Category Content (Semester hours)

<table>
<thead>
<tr>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>0.25</td>
<td></td>
<td>Software design/Tools</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Data structures</td>
<td>0.25</td>
<td></td>
<td>Concepts of programming languages</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Computer organization and architecture/OS/Network</td>
<td></td>
<td></td>
<td>Ethics</td>
<td>0.25</td>
<td></td>
</tr>
</tbody>
</table>

*Dates are estimates. Examinations on dates should follow the university handbook.*
MID TERM AND FINAL
I. Simply answer the following questions (20pts):

a) Define agility and refactoring. Simply describe three values of XP (Extreme Programming). (5pts)

Agility is an effective response to change, effective communication among stakeholders, and a team is organized so that it is in control of the work performed. Refactoring is an iterative refinement of the internal program design and it does not alter the external behavior of the code. Three values of XP are simplicity - means write simple programs; communication - means team communication with customer; and feedback - means customer negotiation.

b) What is software testing? List four types of testing strategies and testing techniques. (5pts)

Software testing is a process in which a program is being tested with the intent of finding errors. Four types of testing strategies are unit test, integration test, validation test, and system test. Four testing techniques are state-based, event-based, top-down, and bottom-up.

c) Describe the difference between traditional unit testing and object-oriented unit testing. (5pts)

Traditional unit testing only focuses on classes and their execution. Object-oriented unit testing includes analyzing logical problem and the strategies include thread-based testing, use-based testing, and cluster testing. It also encounters en encapsulation which is when data is hidden from the user and can not be tested.

d) Define the formal specification language. What are the desired properties of formal methods? Give two examples of formal specification languages. (5pts)

Formal specification language is the mathematical based technique that helps with implementation of systems and software. The desired properties are that it ensures that the customer gets what they want and it helps protect the
III. Please identify the test cases for the following code segment using basis path testing: 
(15pts)

```java
Procedure SaleSalary (n: integer, key : array[1..n] of integer, name: array[1..n] of string, level : integer, personname: string)

var j, salary: double; promotion : double

begin
  salary := 0;
  l := 0;
  personname := null;
  if (i < n && key[i] < 5)
  salary := 400
  else
    salary := 800
  endif

  for j := 1 to n do
    if (key[j] = task[j]) then salary := salary + 100
    promotion := salary * task / key;
    if (promotion < 0.2) then level := 0
    else if (promotion >= 0.2 && promotion < 0.5) then level := 1
    else if (promotion >= 0.5 && promotion < 0.8) then level := 2
    else level := 3
  endif
  endfor

You must have at least three steps – flow graph, cyclomatic complexity number, and independent path.
Cyclomatic Complexity Number
method (1): C
method (3): E - NH₂
\[ 32 - 22 + 2 = 6 \]

Independent Path on next page

Advanced Topic 1
Advanced Topic 2
I. Simply answer the following questions (20pts):

a) Define agility and refactoring. Simply describe three values of XP (Extreme Programming). (5pts)

   - Agility is based on iterative and incremental development.
   - Refactoring involves restructuring existing code without changing external behavior.
   - Extreme Programming is intended to improve software quality, improve responsiveness, and improve productivity.

b) What is software testing? List four types of testing strategies and testing techniques. (5pts)

   - Process of exercising a program with intent of finding errors prior to delivery to the end user.
   - Unit test
   - Integration test
   - Validation test
   - System test

(c) Describe the difference between traditional unit testing and object-oriented unit testing. (5pts)

   - Object-oriented testing begins by evaluating the correctness and consistency of the OOP and OOD models. Traditional unit testing involves individual units of source code.

(d) Define the formal specification language. What are the desired properties of formal methods? Give two examples of formal specification languages. (5pts)

   - Formal specification language is the formal language used in computer science during system analysis, requirement analysis, and system design.
   - Two examples are UML and LOTOS.
III. Please identify the test cases for the following code segment using basis path testing.

(15pts)

Procedure SaleSalary (n: integer, key, task: array[1..n] of integer, name: array[1..n] of string, level: integer, personname: string)

var j, salary: double; promotion: double

begin

    salary := 0;
    l := 0;
    personname := null;
    if (i < n && key[i] < 5)
        salary := 400
    else
        salary := 800
    endif

    for j := 1 to n do
        if (key[j] = task[j]) then salary := salary + 100
        promotion := salary * task / key;
        if (promotion < 0.2) then level := 0
        else if (promotion >= 0.2 && promotion < 0.5) then level := 1
        else if (promotion >= 0.5 && promotion < 0.8) then level := 2
        else level := 3
        endif

    endfor

end

You must have at least three steps – flow graph, cyclomatic complexity number, and independent path.
cyclicomatic complexity number = 
21 - 20 + 2 = \( \boxed{3} \)

IP = \( \boxed{7} \)

P1: 1 - 2 - 3 - 4 - 6 - 7 - 9 - 10, 11
P2: 1 - 2 - 3 - 4 - 5
P3: 1 - 2 - 3 - 4 - 6 - 7 - 8 - 9 - 10 - 12
P4: 1 - 2 - 3 - 4 - 6 - 7 - 8 - 9 - 10 - 12 - 13 - 14
P5: 13 - 15 - 16
P6: 13 - 17 - 18
P7: 13 - 19 - 20

Advanced Topic

Page 5
I. Simply answer the following questions (20pts):


Three values are: (1) Simple programs; (2) Communication; (3) Feedback & Customer feedback.

(b) What is software testing? List four types of testing strategies and testing techniques. (5pts) Software testing: process of executing a program with the intent of finding errors.

**Testing Strategies**
1. Unit Test
2. Integration Test
3. System Test
4. Acceptance Test

(c) Describe the difference between traditional unit testing and object-oriented unit testing. (5pts) Unit testing: broken down due to encapsulation. Focus on classes and their execution. Object-oriented unit testing: concept - top-down. Integration testing: objects are found easily.

Object-oriented unit testing: testing unit is the encapsulated class or object.

(d) Define the formal specification language. What are the desired properties of formal methods? Give two examples of formal specification languages. (5pts)

Formal specification: mathematically based techniques that help with implementation of system and software. Advantages: Ensures that the customer gets what they want. It helps protect programmer from segmentation changes from the customer.

Formal specification:

\[ x \rightarrow y \]

and \( x \rightarrow y \) implies...
III. Please identify the test cases for the following code segment using basis path testing:

(15pts)

```pascal
Procedure SaleSalary (n: integer, key : array[1..n] of integer, name: array[1..n] of string, level : integer, personname: string)

var j, salary: double; promotion : double

begin
  salary := 0;
  i := 0;
  personname := null;
  if (i < n && key[i] < 5)
  then salary := 400
  else
  begin
    salary := 800
  endif

for j := 1 to n do
  if (key[j] = task[j]) then salary := salary + 100
  promotion := salary * task / key;
  if (promotion < 0.2) then level := 0
  else if (promotion >= 0.2 && promotion < 0.5) then level := 1
  else if (promotion >= 0.5 && promotion < 0.8) then level := 2
  else level := 3
  endif
endfor
```

You must have at least three steps – flow graph, cyclomatic complexity number, and independent path.
QUIZ/TESTS
Formal specification language is a mathematical based methodology that provides systematically analysis and description of systems. OCL is one typical example. OCL is based on the set theory and first order logic.
4.

Idle → search → Active

- Select genre(s)
- Select DVD(s)

Checkout

- 4 DVDs for $10
- 10 DVDs for $20
- 30 DVDs for $30

Payment Options
I. Define formal Specification language and give an example with simple explanation why it is a formal specification language. (Advanced Topic 2)

II. Given the descriptions as follows, simply model the system based on the requirements. You may choose one of them.

1. Given a copy machine that takes smart card to start. After the user insert smart card, the user will be requested to input password. If password is validated, the user can start copy and can send the scanned copy to others using email. The menu is shown in the LED. Please define a state diagram using correct UML notations.

2. Given a gas station with 9 stations. Each station has three types of gas (87, 90, and 93). Each station can be paid by credit or cash (inside). The LED is used to display the gallons and total price that currently filling. The receipt can be printed if the customer needs. Please define a state diagram using correct UML notations.

3. Given a snack machine that accepts quarters and bills ($1, $5, and $10). By selecting the items, the user can get a snack with enough payment, and dispense the item and change if the money is more than the item price. The LED will display each step properly. If there is not enough money inserted, the user can either insert more money or get the money back by press the return button on top of the return slot. Please define a state diagram using correct UML notations.

4. Given an online DVD rental system. A customer can search, select and check out a certain number of DVDs providing all are available and the program selected. There are three programs: a) check out 4 DVDs with $10 each month; b) check out 10 DVDs with $20 each month; c) check out 30 DVDs with $30 each month. Please define a state diagram using correct UML notations.

Note:
1) Make sure you work on these questions individually. Two identical works will get ZERO.
2) If you can finish more than one questions, you may get extra points.
Formal specification language is used to accurately model the system.
I. Define formal Specification language and give an example with simple explanation why it is a formal specification language.  

II. Given the descriptions as follows, simply model the system based on the requirements. You may choose one of them.

1. Given a copy machine that takes smart card to start. After the user insert smart card, the user will be requested to input password. If password is validated, the user can start copy and can send the scanned copy to others using email. The menu is shown in the LED.
   Please define a state diagram using correct UML notations.

2. Given a gas station with 9 stations. Each station has three types of gas (87, 90, and 93). Each station can be paid by credit or cash (inside). The LED is used to display the gallons and total price that currently filling. The receipt can be printed if the customer needs.
   Please define a state diagram using correct UML notations.

3. Given a snack machine that accepts quarters and bills ($1, $5, and $10). By selecting the items, the user can get a snack with enough payment, and dispense the item and change if the money is more than the item price. The LED will display each step properly. If there is not enough money inserted, the user can either insert more money or get the money back by press the return button on top of the return slot.
   Please define a state diagram using correct UML notations.

4. Given an online DVD rental system. A customer can search, select and check out a certain number of DVDs providing all are available and the program selected. There are three programs: a) check out 4 DVDs with $10 each month; b) check out 10 DVDs with $20 each month; c) check out 30 DVDs with $30 each month.
   Please define a state diagram using correct UML notations.

Note:

1) Make sure you work on these questions individually. Two identical works will get ZERO.

2) For questions in part II, if you can finish more than one questions, you may get extra points.
This is a screenshot of the application running. This is where we are able to enter the professor's first and last name into the database. The second screenshot shows the computer science professor information inserted.

<table>
<thead>
<tr>
<th>My Contacts</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dr. Venkata Atluri</td>
</tr>
<tr>
<td>2</td>
<td>Dr. Alakananda Bandyopadhyay</td>
</tr>
<tr>
<td>3</td>
<td>Mr. Nelson Barnes</td>
</tr>
<tr>
<td>4</td>
<td>Mr. Willie Bossie</td>
</tr>
<tr>
<td>5</td>
<td>Dr. Jian Fu</td>
</tr>
<tr>
<td>6</td>
<td>Dr. Yujian Fu</td>
</tr>
<tr>
<td>7</td>
<td>Dr. Jay Gangasani</td>
</tr>
<tr>
<td>8</td>
<td>Dr. Muhammad Ghanbari</td>
</tr>
<tr>
<td>9</td>
<td>Dr. Xiang Xhao</td>
</tr>
</tbody>
</table>
VIII. Validation Methodology

For testing our software, we used the basis path testing method; basic path testing is a white-box testing technique [5]. White-box testing is a test-case design philosophy that uses the control structure described as a part of component-level design to derive test case [5]. The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use it to perform testing on each path of the code [5]. The notation we used to represent are testing method is the flow graph notation. The flow graph illustrates the individual path of each statement of the code and show each path the code can take.

This diagram below shows the flow graph of a segment code that we used in testing our program:

```
1) public void onCreate(Bundle savedInstanceState) {
2)     super.onCreate(savedInstanceState);
3)     setContentView(R.layout.add_activity);
4)     btn_save=(Button)findViewById(R.id.save_btn);
5)     edit_first=(EditText)findViewById(R.id.first_editTxt);
6)     edit_last=(EditText)findViewById(R.id.last_editTxt);
7)     if(isUpdate=getIntent().getExtras().getBoolean("update")
8)         {
9)             id=getIntent().getExtras().getString("ID");
10)            fname=getIntent().getExtras().getString("Fname");
11)            lname=getIntent().getExtras().getString("Lname");
12)            edit_first.setText(fname);
13)            edit_last.setText(lname);
```
Method 1: # of Regions = 8

**Identifying Independent Paths**

IP 1
IP 2
IP 3
IP 4
IP 5
IP 6
IP 7
IP 8

Generate Test Case: There will be a minimum of 8 test cases for this segment of code.

IX. Conclusion

In conclusion, the user interface allows the user to enter and view the professor's name. Soon, students will have the ability to view their professor's phone number and email address. Due to complications, we were unable to include this information within this report. After implementing the contact information, we will also include the office hours for each professor within the Computer Science department. The FindIt application will allow the user to select each professor and view the professor's information. FindIt is still a work in progress. We are constantly developing ideas on how to implement this application. As stated before, we have a strong desire for this application to be a useful tool throughout the Computer Science department.

References
This image, like all the others with parameters that were too large for the target footprints, is messy and unclear. You can vaguely pinpoint a target, but it's definitely not clear like the maps that had well estimated parameters. Working on this project showed us two things. First, Matlab is hard. The second being, Matlab is easy, once you take the time to understand it. As said before, we felt that working with Matlab was a difficult and big disadvantage. But working on it continuously has showed us how much fun and interesting it really is.

Validation Methodology

For testing our software, we used the basis path testing method which is called white box testing technique. This basic path testing is a test-case design philosophy that uses the control structure that is described as a part of component-level design to derive test case. The basis path method enables the test-case designer to achieve a logical complexity measure of a procedural design and use it to perform testing on each path of the code. The flow graph notation was used and it basically illustrates the individual path of each statement of the code and shows each path the code can take.

This section is where the white-box testing took place.

1. function Image_CBL = ContrastBoxLevel(InputImage, PatchDs);
2. M = PatchDs(1);
3. N = PatchDs(2);
4. P = PatchDs(3);
5. Q = PatchDs(4);
6. II3 = soldiers(1, II1, InputImage, II2);
7. II4 = II3(:, 1:N2);
8. II5 = II3(:, (N1-N2+1):N1);
9. II0 = soldiers(2, II4, II3, II5);
10. Image_CBL = zeros(M1, N1);
11. for im = (M+N):(P+Q);
12.  u1 = im - MN;
13.  u2 = im + PQ;
14.  for in = (N2+1):(N1+N2);
15.  v1 = in - N2;
16.  x2 = in + N2;
17.  W = II0(u1:x2, v1:v2);
18.  Image_CBL((N-M), (Q-P)) = max(max(W));
19.  End
Cyclomatic Complexity #

Method 3: N-E+2 = 20-3+2 = 20 edges

Identifying Independent Paths

IP 1,

IP 2,

IP 3,

IP 4,

IP 5,

IP 6
IP 7,
IP 8,
IP 10
IP 13
IP 17

Generate Test Case: There will be a minimum of 8 test cases for this segment of code.

Conclusion

This project is interesting and pushes us to make educated guesses, while helping us to better understand MATLAB. The proposed method that avoids complications due to target or clutter variability by employing general size, connectivity, and motions criteria to is called ATDT. One feature is to differentiate the hot and cold spots in an image. This has motivated us to basically seek out for companies that has this technology involved within a job description. Northrop Grumman is one of the companies that has technology to conduct for a certain job.
IX. Validation Methodology

Our code was validated by the system testing methodology. Prior to the system testing, we conducted integration testing, and combined individual stages together and tested them as a group. The purpose of the integration testing was to verify the functionality, performance and reliability of the proposed coding. Our Integration method of choice was the Big Bang Method. The developed stages were coupled together to form a complete software system and then we used them for integration testing. We chose this method because it saved a tremendous amount of time. [3]

After completion of the basic integration testing, we used the system testing because it can be performed on the entire coding project at once. Another benefit of using the system testing method is the fact that when system testing, the entire design is tested and the behavior of the code is tested as well. [4]

For future code development, we are looking into using Soak testing also know as endurance testing. The use of soak testing is important to our project, because of the nature of work that the code will be used for. The code must be reliable and able to sustain a continuous load of work from daily use. [4]
Basis Path Testing Results

Cyclomatic Calculation
Method A: Regions 1
Method B: 0+1=1
Cyclomatic Number: 1
Predicate Nodes: 0
Edges: 11-12+2=-1
Independent Path: 1
IP1: 1-2-3-4-5-6-7-8-9-10-11-12