

Mimi Zhang

by Mimi Zhang

FILE	17847_MIMI_ZHANG_MIMI_ZHANG_227501_1194778172.DOCX (461.94K)		
TIME SUBMITTED	20-JAN-2020 05:03AM (UTC+0000)	WORD COUNT	4365
SUBMISSION ID	118719019	CHARACTER COUNT	24196

Task 1

Research Report about TDD, BDD and FDD in Agile Software Engineering

Principles

Student Name: Mimi Zhang

Student Number: 1904733

ACCA7000

UWTSD

Instructor: Dr Tim Bashford & Dr Mike Dacey

20-Jan-2020

Abstract

Agile development is a new method of software development, which has attracted more and more attention. TDD, BDD and FDD play an important role. This paper introduces TDD, BDD and FDD respectively, and explains their advantages and disadvantages as well as their differences. Finally, it explains their best application scenarios.

Content

Abstract.....	1
Content	2
1. Test-Driven Development	3
1.1 The definition of TDD	3
1.2 The work process of TDD	3
1.3 Advantages and disadvantages of TDD.....	4
2. Behaviour-Driven Development	5
2.1 The definition of BDD	5
2.2 The work process of BDD	5
2.3 Advantages and disadvantages of BDD.....	6
3. Feature-Driven Development	7
3.1 The definition of FDD	7
3.2 The work process of FDD	7
3.3 Advantages and disadvantages of FDD	8
4. Conclusion.....	8
4.1 Difference between TDD, BDD and FDD	8
4.2 Best-applied scenarios	9
Reference	10

1. Test-Driven Development

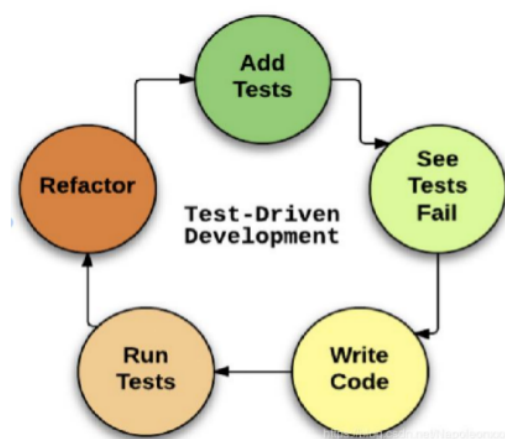
1.1 The definition of TDD

TDD is an important method in agile development. It means writing unit tests before developing code, and using test code to decide what code to write. The whole process drives the development of software through testing, which places higher requirements on testers [1].

1.2 The work process of TDD

- 1) Writing test. Write functional tests first, and then describe the new features of the application from the user's perspective.
- 2) Check and confirm the test. After a functional test fails, find a way to write code to make it pass. At this point, use one or more unit tests to define the effect you want your code to implement, and ensure that you write one unit test for each line in your application code.
- 3) Write the product code and test it. After a unit test fails, write a minimum number of application code to pass the unit test. Sometimes, we have to go back and forth many times before the second and third steps, until we feel that functional testing has progressed.
- 4) Refactor the product code. Run the functional test again to see if it passes or is in progress. This step may prompt us to write some new unit tests, code, etc.

Repeat the above steps when new requirements and functions need to be developed [2]. The process is as follows:



1.3 Advantages and disadvantages of TDD

1.3.1 Advantages of TDD

- 1) High code coverage. System test and integration test are not easy to cover some code details, and it is difficult to achieve high code coverage; the test written by Test-Driven development is usually unit test, which is written by developers. For a single module, it is easy to achieve test coverage for each scenario.
- 2) Fast test speed. Unit testing accelerates testing. Manual testing without unit testing can take a lot of time. To be precise, fast feedback is the benefit of unit testing.
- 3) Timely feedback. Testers add tests to return functions or errors. The cycle is too long and the feedback is not timely, which is not conducive to the development enthusiasm. Test-Driven development testing continues, focusing on one fine-grained issue at a time. The test shows that the problem has been solved, and it is easy to get a sense of achievement by feedback development in time. In addition, if the development team is large or the modules are frequently modified, errors may be damaged or reappeared due to the untimely regression; Test -Driven development and testing will continue without untimely regression.
- 4) Easy to locate errors. It is difficult to focus and locate problems by using system tests or integration tests to regress functions or bugs. Test-Driven development covers only one scenario per test. Once the test fails, you can know the problem without debugging.
- 5) Tests are documents. Unit tests are documents that don't lie.
- 6) Test-Driven Development is the limitation of development. If the test is well written, the program design will not be too bad, the coupling of modules will not be too strong, and there will not be too many bugs.
- 7) The biggest advantage of TDD is that it has complete unit testing and provides a protection network, making it easier to change requirements or improve code.
- 8) The process of writing tests can help you understand and refine requirements, decompose problems, and then cover them one by one and solve them step by step;
- 9) The benefits of Test-Driven replacement for debugging, functional completion, or bug fixes can also be regressed.
- 10) Identify requirements in advance. Writing tests at the beginning can help us think about requirements and their details in advance.

1.3.2 Disadvantages of TDD

- 1) In some cases, because the product code is complete, there are many business associations and coupling between objects. If you write it again, it's easy to fall into a mindset and find no mistakes. Although interfaces can be added to achieve separation, the process of building a test context becomes more complex.
- 2) Tests are also code. Tests themselves may not be easy to maintain when requirements change frequently, or during the development testing phase, or when the program structure changes frequently.
- 3) This places too much emphasis on unit testing. Imagine that if unit test coverage was found to be inadequate before the product was released, this would allow all employees to write test cases [3].

2. Behaviour-Driven Development

2.1 The definition of BDD

Behavior-Driven Development implements an application by describing its behavior from a stakeholder perspective [4]. BDD defines stakeholders as anyone who cares about what we do, whether they're the people we're trying to solve or the people who help solve the problem. From the perspective of BDD, even people in the delivery team are stakeholders. In this context, behavior refers to the behavior of the project from the perspective of stakeholders.

2.2 The work process of BDD

The BDD process can be divided into three phases: discovery, development, and automation, in which acceptance criteria are transformed into acceptance testing, and then automation [5].

1) Discovery

At this stage, the product owner or product manager first creates the acceptance criteria as part of the story or project.

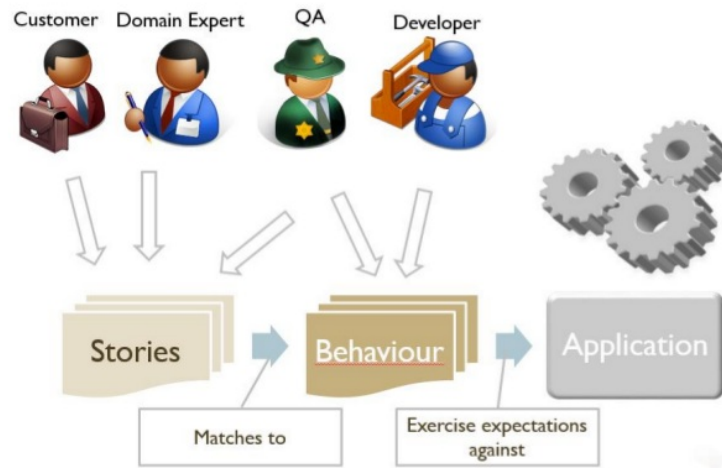
2) Development

Near implementation, the development phase will incorporate acceptance criteria to create acceptance tests. In the development phase, ambiguity is resolved by transforming the solution into specific acceptance tests.

3) Automation

The automation phase automates acceptance tests so that they can run continuously and verify new behavior.

The whole test flow of BDD is as shown in the figure:



2.3 Advantages and disadvantages of BDD

2.3.1 Advantages of BDD

- 1) From the perspective of requirements, BDD and requirements instantiation make clear the definition of requirements, unify the understanding of users, product managers, development and testing, make everyone stand on the same level, and make R & D more efficient.
- 2) From the perspective of testing, requirements are testing, and product requirements are testing requirements. Requirements can be executed, that is, in one step, requirements are converted into automated test scripts. The developed functions can be verified automatically at any time.

2.3.2 Disadvantages of BDD

- 1) Functional testing is carried out, but the results are not necessarily the quality of the underlying work. Unless you're also doing test driven development, you risk building the right code in the wrong way, which can be difficult to maintain or extend in the future.
- 2) There are fewer software tools to support BDD.

3. Feature-Driven Development

3.1 The definition of FDD

Function driven development (FDD) is agile development. This framework enables software development to make progress in functionality [6]. However, the features in FDD are not necessarily the product features we understand. They are more like use cases in scrum.

3.2 The work process of FDD

FDD design follows five steps, each built around a discrete "properties" project. Its life cycle is as follows:

1) Build an overall model

Under the guidance of experts, demanders work with developers. According to the introduction, the demand side and the development side will generate the initial model on this basis, then group the detailed discussion stage, draw the outline of the model, and then enrich the previous initial model.

2) Generate function list

After you create the initial model, you start building a list of complex features. The function list is the occurrence of action actions. Each action action is caused by an object, for example: calculate the total number of orders; add, delete, modify and check some data. It can be seen that this function is not a general function, but a function that occurs around the object model.

3) Planning by function

The next step is to sort the main feature sets and assign them to the owners. At this point, programmers will enter the corresponding programmer group according to the classes they are responsible for in the global object model.

4) Design by function

After the programmer group receives the feature set, 4-5 programmers meet. Then, the team leader selects a small number of functions suitable for completion within 2 weeks, and then performs "design by function" and "build by function". The corresponding classes and functions are assigned by the team leader. Finally, the functional team designed a detailed sequence diagram and wrote the class and method logic.

5) Generate by function

Finally, the function building phase refers to the compilation, debugging and testing of the code.

3.3 Advantages and disadvantages of FDD

3.3.1 Advantages of FDD

- 1) Simple five step process makes development faster.
- 2) The team continues to drive the product forward successfully.
- 3) Teams quickly pass predefined development standards [7].

3.3.2 Disadvantages of FDD

- 1) Due to the high degree of customer participation of FDD, the team culture is easily influenced by its customer culture.
- 2) Because the preliminary work of FDD may be very long, it is easy for people to become a heavy method exhibition with agile flag.

4. Conclusion

4.1 Difference between TDD, BDD and FDD

- 1) Secrets and models. The TDD and BDD processes begin business analysis by recording stories on cards. FDD uses features and performs domain traversal, and constructs a comprehensive domain object model, so that feature groups can generate better designs for each group of features [8].
- 2) Code ownership. TDD and BDD encourage collective ownership of code, and anyone can add or modify code as needed. Instead, in FDD, the entire development team has collective ownership of the code. When collective validation is needed, such as the design of software architecture or the construction of user interface, FDD combines class owners with functional groups and reviews to meet the needs.

3) Testing. TDD and BDD use two person pair programming to perform walkthroughs and informal reviews continuously at the design and code levels. FDD advocates the use of structured formal review techniques. In addition, the correctness of TDD and BDD is defined by running unit and function tests. In FDD, unit testing is part of the build by function process. FDD does not define the formal level of participation in testing, it is more suitable for the master programmer to decide what to do.

4) Project tracking. TDD and BDD allow project managers to determine project tracking methods, encourage less data collection, and encourage the use of large charts. The feature item tracking method in FDD describes the method with small workload and accurate project progress measurement, which provides data for building various progress charts.

In summary, FDD organizes its work based on business feature slices and focuses on the highly iterative delivery process of these features. From this point of view, BDD inherits many advantages of FDD, but BDD goes further. It emphasizes highly automatic verification and communication with business users, and models the correct system through examples. In short, BDD is a mixture of the advantages of FDD and TDD.

4.2 Best-applied scenarios

4.2.1 TDD

TDD is a process of writing and running tests, which supports high test coverage. Test coverage is the percentage of automated test code, so the larger the number, the better. TDD also reduces the error rate that would otherwise be difficult to track. TDD is suitable for scenarios with unstable project requirements, which requires many subsequent changes and corrections. However, if your project has a complex business model and internal modules are interdependent, don't use TDD because it makes writing test code for programmers onerous [9].

4.2.2 BDD

In a broad sense, BDD solves many problems of TDD. The BDD test uses pure language. Tests such as reading sentences can solve communication problems, so testers and developers will be able to better understand users' needs. However, there is little difference between BDD and TDD in the scope of application.

4.2.3 FDD

1) Domain object modeling

It is a form of object decomposition, mainly including the construction of class diagram, which is used to describe the types of important objects in the problem field and their interrelations. It provides an overall framework for system design, so that the system can be developed in accordance with the feature iteration increment.

2) Develop according to characteristics

Develop and track the process according to a list of small functions that are valuable to customers. FDD decomposes the demand problem into small problems that can be solved, and decomposes each problem into functional requirements of hierarchical list, that is, characteristics. Then, design and implement each feature. Once the functional features of the system are identified, they are usually used to drive and track the development process in the FDD method [10].

3) Class (code) ownership

4) Feature group

5) review

6) Build regularly

7) Configuration management

8) Visibility progress report

Reference

[1]Mugridge, R.. Test driven development and the scientific method[P]. Agile Development Conference, 2003. ADC 2003. Proceedings of the,2003. [Accessed:Nov. 15, 2019].

[2]Swamidurai, Rajendran,Dennis, Brad,Kannan, Uma. Investigating the impact of peer code review and pair programming on test-driven development[P]. SOUTHEASTCON 2014, IEEE,2014. [Accessed:Dec. 15, 2019].

[3]Torchiano, M., Sillitti, A.. TDD = too dumb developers? Implications of Test-Driven Development on maintainability and comprehension of software[P]. Program Comprehension, 2009. ICPC '09. IEEE 17th International Conference on,2009. [Accessed:Dec. 08, 2019].

[4]Juan Qiao, Smith, M., Miller, J.. On Moving Test-Driven Development from the Business World into a Biomedical Engineering Environment[P]. Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on,2007. [Accessed:Jan. 08, 2020].

[5]Gupta, A., Jalote, P.. An Experimental Evaluation of the Effectiveness and Efficiency of the Test Driven Development[P]. Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on,2007. [Accessed:Jan. 09, 2020].

[6]Hamill, P., Alexander, D., Shasharina, S.. Web Service Validation Enabling Test-Driven Development of Service-Oriented Applications[P]. Services - I, 2009 World Conference on,2009. [Accessed:Jan. 18, 2020].

[7]Aniche, M.F., Gerosa, M.A.. Most Common Mistakes in Test-Driven Development Practice: Results from an Online Survey with Developers[P]. Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on,2010. [Accessed:Jan. 11, 2020].

[8]Markovic, D., Jovanovic, Z., Pesovic, U., Randic, S.. Test-Driven Development (TDD) in development of measure application and transducers services[P]. Telecommunications Forum (TELFOR), 2011 19th,2011. [Accessed:Jan. 13, 2020].

[9]Williams, L., Maximilien, E.M., Vouk, M.. Test-driven development as a defect-reduction practice[P]. Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on,2003. [Accessed:Jan. 15, 2020].

[10]]Ioan Lazăr,Simona Motogna,Bazil Pârv. Behaviour-Driven Development of Foundational UML Components[J]. Electronic Notes in Theoretical Computer Science,2010,264(1). [Accessed:Jan. 15, 2020].

Task 2

**Research Report about Importance of High Cohesion, Loose Coupling, Data Hiding
and Information hiding in software Development**

Student Name: Mimi Zhang

Student Number: 1904733

ACCA7000

UWTSD

Instructor: Dr Tim Bashford & Dr Mike Dacey

20-Jan-2020

Abstract

In software programming, there are couples of concepts using for increase understand ability and reusability. Some of developer consider programming should be high cohesion and lose coupling. They use data hiding and information hiding to protect the information get changed when they shouldn't be changed.

This article focuses on research the importance of cohesion, coupling and data hiding using in software development. Talk about the definite of those concepts. Try to use those technics to increase software robustness and reliability. Give example to explain how they are working in development.

Content

Abstract.....	13
Content	14
1. High cohesion and loose coupling	15
1.1 Cohesion.....	15
1.2 Coupling	15
1.3 Cohesion and Coupling using in software development	16
2. Data hiding and information hiding	17
2.1 Data hiding	17
2.2 Information hiding	17
2.3 Data hiding and information hiding using in software development	17
3. Conclusion.....	18
Reference	18

1. High cohesion and loose coupling

1.1 Cohesion

Cohesion means the degree between the elements which inside module. In another word, that is a measure of the closeness between the module of class and data inside module. If the class's model and data closer, we can consider the module is high cohesion. So, developer usually us "high cohesion" and "low cohesion" to measure the relationship between the elements which inside module. We prefer high cohesion module to increase the performance of software. High cohesion module enhances software robustness reliability, reusability, and understand ability. That will make developer easier to understand programming, increase development efficiency, reduce maintenance time. Make the coding more stable by avoid data get changed when they shouldn't be. Get more easier to implement new foundation when we develop [1].

High cohesion: It depends on the relationship between classes. High cohesion means that the relationship between them should be simple and clear, without strong relationship. Otherwise, there will be running problems. The operation of one class affects other classes. High cohesion is recommended for module design because of its robustness, reliability, reusability and readability. High cohesion is a relatively independent approach that reduces external impact. Low coupling is relatively method dependent and can be used to build programs. Cohesion is similar to building blocks, and coupling is similar to building houses. High cohesion and loose coupling is a more efficient and personalized way to build a house [2].

1.2 Coupling

Coupling is another concept in computer programming. That is the independence between the modules. Developer usually contrasted that with cohesion, they prefer to develop high cohesion and loose coupling computer systems [3]. As the same way, low coupling system can reduce the maintenance time and easier to develop. That is the well structure when people coding program.

Loose coupling: In object-oriented programming, the object itself is cohesive, it saves its own data, completes its own operation, and presents its own state and behavior to the outside world. However, without absolute self-reliance, opening up is also necessary! An object usually needs to deal with other objects, including not only knowing the state of other objects, but also depending on the behavior of other objects. Once this happens, we call it an object that depends on another object. As long as there is a one-to-one relationship between two objects, we call it the coupling between two objects. Coupling degree is coupling degree, that is, the degree of interdependence [4].

1.3 Cohesion and Coupling using in software development

High cohesion and loose coupling systems are widely used in software development, especially in object-oriented programming [5]. That can make software system get high readability and maintainability. There are couples of ways to build a strong cohesion and weak coupling system.

Coupling and cohesion are two qualitative criteria of module independence. When the software system is divided into modules, we try to achieve high cohesion and low coupling, improve the independence of modules, and lay the foundation for the design of high-quality software structure [6].

How to reduce coupling of system.

- 1)Developer should reduce class inheritance. Try to use more interface and hiding the details of implementation.
- 2)The functionalization of module should be got simple to reduce the opportunity of that module called by other modules.
- 3)Make sure the definition only appears one place. Avoid repeat definition which will cause system declined.
- 4)Try to use as much “private” as we can when we declare class method. Try to reduce public class.
- 5)Try to use global as variables as we can.
- 6)Try to use multi-purpose design patterns like MVC. That can reduce the logic and lower coupling
- 7)Do not use “hard code” to coding the program. Avoid using SQL statement to operate database.

How to increase cohesion.

- 1)Minimize interface expose of module, try to reduce dependency relationship.
- 2)If the interface not changed, we should not affect other modules when we modify inside of modules.
- 3)If we delete one module, not related parts should not be affected.

In short, in the actual application environment, high cohesion does not mean that a certain code only realizes one function, but is determined according to the actual needs of the project. Generally, if there are more than 100 lines of code to implement a function, the function can be decomposed into several sub modules as needed. In this way, it has more advantages in debugging and function updating. Loose coupling is a concept that needs to be weighed repeatedly in practical application. Loose connection brings convenience in use, but also increases the difficulty of design. It is in the contradiction of high cohesion, and in the development work, an excessive urination design can maximize the importance of these two principles [7].

2. Data hiding and information hiding

2.1 Data hiding

The concept of data hiding comes from object-oriented programming, which is used to prevent unauthorized access to ensure the security of class members. Therefore, data hiding can protect data members from manipulation or intrusion from any other source. As we all know, the data in the program is sensitive and unstable, so we must be careful when operating, otherwise it will cause errors and destroy the integrity of the data [8].

2.2 Information hiding

Information hiding in object-oriented programming refers to hiding the key parts of an object to expose its internal implementation and higher abstraction. A simple and easy to understand example is that when we are playing a four chess game, we only need to enter the number of columns, T key and B key, and do not need to know what is inside.

2.3 Data hiding and information hiding using in software development

Data hiding:

By limiting the interdependency between software components, data hiding reduces the complexity of the system and increases the robustness [9].

Information hiding:

A common use of information hiding is to hide the physical storage layout of data so that changes are limited to a small part of the entire program when changing data. In object-oriented programming, information hiding (through type nesting) reduces the risk of software development by transferring the dependence of code on uncertain implementation (design decision) to a well-defined interface. The client of the interface performs operations only through it, so if the implementation changes, the client does not have to change [10].

The goal is to realize the possibility of change: you can improve the internal mechanism of a component without affecting other components, or you can replace it with other components that support the same common interface. Information hiding also protects the integrity of components

by preventing users from setting their internal data to an invalid or inconsistent state. Another advantage of information hiding is that it reduces the complexity of the system by limiting the interdependence between software components.

Information hiding is an effective standard to divide any device, software or hardware into functional modules. Information hiding provides flexibility. This flexibility allows programmers to modify the functions of computer programs during normal evolution, because computer programs have been changed to better meet the needs of users. When a computer program is carefully designed to use the principle of information hiding to decompose the source code solution into modules, evolutionary changes are much easier, because changes are usually local rather than global.

Because data hiding only hides the private fields of data structure, information hiding may refer to the encapsulation implementation details of communication only through interfaces.

3. Conclusion

In a word, high cohesion and low coupling are the basic principles of programming. In programming, we should always keep in mind, weigh repeatedly, increase the center of gravity of the function as much as possible, make the function of each module tend to simplify, at the same time, reduce the dependency between modules as much as possible, and make the function call more convenient. Information hiding and data hiding are prerequisites for good code. Global variables can continue to change throughout the project, while local variables can continue to play a role in the scope of subclasses. They complement each other to produce efficient and secure code.

Reference

[1]Li Hui, He Yu, Jiang Zhihong, Huang Yuancan, Huang Qiang. High-cohesion and low-coupling integrative joint for space manipulator[P]. Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on,2009.

[2]. Information Technology - Information and Data Encoding and Encryption; Researchers at University of Jinan Have Reported New Data on Information and Data Encoding and Encryption (Reversible Data Hiding Algorithm in Fully Homomorphic Encrypted Domain)[J]. Computers, Networks & Communications,2019.

[3]. Information Technology - Information and Data Encoding and Encryption; Studies from Nanjing University in the Area of Information and Data Encoding and Encryption Reported (A Source Hiding Identity-Based Proxy Reencryption Scheme for Wireless Sensor Network)[J]. Computers, Networks & Communications,2018.

[4]. Information Technology - Information and Data Security; Researchers from Harbin Institute of Technology Report Recent Findings in Information and Data Security (Metasurface Hologram for Multi-Image Hiding and Seeking)[J]. Computers, Networks & Communications,2019.

[5]Peter Pangestu,Dennis Gunawan,Seng Hansun. Images with hidden information data set for information retrieval usage[J]. Data in Brief,2019,26.

[6]Xuanyu He,Wei Zhang,Haifeng Zhang,Lin Ma,Yibin Li. Reversible data hiding for high dynamic range images using edge information[J]. Multimedia Tools and Applications,2019,78(20).

[7]Dongdong Hou. Emerging Applications of Reversible Data Hiding[C]. Proceedings of The 2nd International Conference on Image and Graphics Processing(ICIGP 2019)Workshop The 5th International Conference on Virtual Reality(ICVR 2019).2019:108-112.

[8]J N Liu. A High Efficient FEMD-Based Data Hiding Algorithm[C]. Asia Pacific Institute of Science and Engineering.Proceedings of 2019 3rd International Conference on Computer Graphics and Digital Image Processing (CGDIP 2019).

[9]Jiin-Chiou Cheng,Wen-Chung Kuo,Bo-Rong Su.Data-Hiding Based on Sudoku and Generalized Exploiting Modification Direction[J].Journal of Electronic Science and Technology,2018,16(02):123-128.

[10]Parah Shabir A,Ahad Farhana,Sheikh Javaid A,Bhat G M. Hiding clinical information in medical images: A new high capacity and reversible data hiding technique.[J]. Journal of biomedical informatics,2017,66.

FINAL GRADE

75/100

GENERAL COMMENTS

Instructor

Task 1 report (35%). 32

Task 1 use of git (15%) 9

Task 2 report (20%) 12

Task 2 code implementation (30%) 22

Comments

=====

BDD and FDD are described in good detail and some scenarios are supplied. Quite nice discussion of BDD.

Would like to have had more examples of the application of cohesion and coupling to better illustrate these concepts. Some of the analogies were a little vague.

Couple of errors in the code (bounds checking required), more testing is needed.

Some testing provided.

Reasonable git usage, no branches and comments are not sufficiently descriptive

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20
